

TOWARDS NATURAL AND ROBUST HUMAN-ROBOT INTERACTION USING SKETCH AND SPEECH

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Danelle Christine Shah

January 2012

© 2012 Danelle Christine Shah
ALL RIGHTS RESERVED

TOWARDS NATURAL AND ROBUST HUMAN-ROBOT INTERACTION USING SKETCH AND SPEECH

Danelle Christine Shah, Ph.D.

Cornell University 2012

For centuries, we have dreamt of intelligent machines that could someday co-exist with humans as autonomous agents, working for, with, and sometimes even against us. Since Karel Čapek's play, *R.U.R. (Rossum's Universal Robots)* was written in 1920 [1], robots have permeated science fiction books, movies and television, giving rise to famous characters such as Robbie in *I, Robot* [2], Johnny 5 in *Short Circuit* [3], and C-3PO in *Star Wars* [4]. However, the fields of robotics and artificial intelligence are still a long way off from producing fully-autonomous machines like Rosie from *The Jetsons* [5] that can behave and interact as humans do. Today, getting computer agents to perform even the simplest of tasks requires designing an interface that is able to translate what the human *wants* into what the computer can *do*. Traditionally, this has been accomplished by constraining human users to communicate in a specific and unambiguous way, such as pressing buttons or selecting options from a menu. This type of interaction is rigid and unnatural, and is far from how humans communicate with one another.

In recent years, there has been growing interest in the development of more natural and flexible human-robot interfaces, allowing humans to communicate with machines using means such as speech, drawing, gesturing, etc. These methods are still in their infancy, and while they offer more human-like interaction with computers, ensuring that the user's intentions are correctly inter-

puted places limits on the flexibility of expression allowed by such systems. For example, despite recent advances in speech recognition technology, natural language interfaces are still largely confined to simple applications in which the speaker's intentions are disambiguated through the use of pre-defined phrases (e.g., "Call home"), or do not need to be interpreted at all, such as for data entry or speech-to-text processing.

In this dissertation, a number of algorithms are proposed with the aim of allowing users to naturally communicate with a semi-autonomous robot while placing as few restrictions on the user's input as possible. The methods presented here reside in the domains of sketch and speech, which are flexible in their expressiveness and take advantage of how humans communicate with each other. The application considered in this work is mobile robot navigation, i.e., instructing a semi-autonomous robot to move to a specific location within its environment, where it will presumably undertake some useful task. By allowing the user to use speak and sketch naturally, the burden of recognition is shifted from human to machine, allowing the user to focus attention on the task at hand. This dissertation develops a probabilistic framework for sketch and speech recognition, the model for which is learned from training data such that recognition is accurate and robust. It also introduces a method for qualitative navigation, allowing the human user to give navigation instructions using an approximate sketched map. These approaches encourage the robot to understand how humans communicate, rather than to force the human to conform to a communication structure designed for the robot, taking a small step towards truly natural human-robot interaction.

BIOGRAPHICAL SKETCH

Danelle was born and raised in Upstate New York. She attended the State University of New York at Buffalo from 2002–2006, where she double-majored in Mechanical Engineering and Aerospace Engineering. She was an active member of the University at Buffalo chapter of the American Institute of Aeronautics and Astronautics, for which she served as Treasurer from 2003–2005 and President from 2005–2006. She was also member of several academic honor societies, including the Phi Eta Sigma National Honor Society, Sigma Gamma Tau National Aerospace Engineering Honor Society, and Tau Beta Pi National Engineering Honor Society. From 2004–2006, Danelle worked as an Engineering Co-op at Moog, Inc. in East Aurora, NY in the Design, Analysis, and Product Engineering Groups. In 2005, she was awarded the Barry M. Goldwater Scholarship, a national award granted to college sophomores and juniors who intend to pursue careers in the fields of science, mathematics, and engineering. Danelle graduated *Summa Cum Laude* from the University at Buffalo in 2006 and was awarded the SUNY Chancellor's Award for Student Excellence.

In 2006, Danelle began her graduate studies at Cornell University in the Mechanical and Aerospace Engineering Department. She joined the Autonomous Systems Laboratory under Dr. Mark Campbell, where her research focused on the interaction between humans and intelligent machines, and the development of more natural communication interfaces for human-robot interaction. Danelle received a National Defense Science and Engineering Graduate Fellowship in 2007, and has had the opportunity to present her work at international conferences in Seattle, Montréal, Taipei and Shanghai.

Danelle married her husband, Ashish, in 2007 and they are expecting their first child in September, 2011.

This thesis is dedicated to my best friend and husband, Ashish, for his infinite
patience, love, and support.

ACKNOWLEDGEMENTS

First and foremost, I want to thank my research advisor and committee chair, Dr. Mark Campbell. He has been an amazing mentor and role model, and has been a great source of encouragement and motivation. He has not only led me in exploring an unfamiliar field of human-robot interaction, but has also given me the opportunity to explore the world. I am truly grateful for his patience, guidance and advice over the past five years.

I also want to express my sincere appreciation to my other committee members, who have helped guide me and focus my work. I could always depend on Dr. Hadas Kress-Gazit to have her door open to discuss research or class work, and for career and general life advice. Dr. Hod Lipson's thought-provoking questions helped lead me towards my most recent work in multimodal interaction, and has pushed me to think a little more outside of the box.

To the past and present members of the Autonomous Systems Lab, I am so grateful to have had such wonderful colleagues and friends these past five years. Thank you especially to Nisar Ahmed, Jon Schoenberg, Mark McClelland, Cesar Rivadeneyra, Tauhira Hoossainy, Joe Schneider, Jason Hardy, Frank Havlak and Frédéric Bourgault for all your help and advice both in and out of the lab. Thank you also to Daniel Lee, Chuck Yang, Jason Moran, Pete Moran and Eric Sample, without whom I may have never gotten a robot to move.

I'd like to thank all those who have helped me in countless ways throughout my journey at Cornell: the entire MAE faculty and staff, the Air Force Office of Scientific Research and Department of Defense for funding my work, and our research collaborators at the Air Force Research Laboratory, Carnegie Mellon University, Massachusetts Institute of Technology, George Mason University, and University of Pittsburgh.

My sincerest thanks goes to my friends and family for a lifetime of support and encouragement. I am eternally grateful to my parents, who have sacrificed their time, money, and sanity to give me everything I've needed to succeed. Without their love and support, I would certainly not be the person I am today. I cannot express enough my appreciation for my husband, Ashish, who has laughed, cried, struggled and celebrated through everything with me. Lastly, I want to thank my son, Arya, for giving me the motivation I needed to finish this thesis, and for keeping me company these past eight months. I can't wait to meet you.

TABLE OF CONTENTS

Biographical Sketch	iii
Dedication	iv
Acknowledgements	v
Table of Contents	vii
List of Tables	ix
List of Figures	x
1 Introduction	1
1.1 Natural Robot Interface via Free Sketch Recognition	1
1.2 Qualitative Navigation Strategies	4
1.3 Multimodal Communication for Robot Control	7
1.4 Contributions	9
2 A Sketch Interface for Robust and Natural Robot Control	10
2.1 Probabilistic Sketch Definition	12
2.2 Learning the Variable Duration Hidden Markov Model	15
2.2.1 Observation Feature Set	17
2.2.2 Gesture Classification	19
2.2.3 Stroke Classification	20
2.2.4 Interstroke Classification	22
2.3 Multi-Stroke Sketch Recognition	24
2.3.1 Recognition Algorithm	24
2.3.2 Modified Likelihood Using Search Heuristic	26
2.4 Experimental Design	29
2.4.1 Sketch Command Interface	32
2.4.2 Waypoint Command Interface	34
2.5 Experimental Results	36
2.5.1 Mission Objectives	36
2.5.2 Subjective Operator Feedback	38
2.6 Chapter Conclusions	42
3 A Qualitative Path Planner for Robot Navigation Using Human-Provided Maps	44
3.1 Navigation Using a Qualitative Map	46
3.1.1 Concept and Architecture	46
3.1.2 Performance Metrics	49
3.2 Qualitative Path Planner (QPP)	50
3.2.1 Waypoint Optimization using Potential Energy	51
3.2.2 Waypoint Extraction using Voronoi-Delaunay Graph	54
3.2.3 Estimation of Sketch Transformation	58
3.2.4 Extensions Addressing Implementation Issues	60
3.3 Experimental Results	65

3.3.1	Illustrative Examples	65
3.3.2	Sensitivity to Sensor Limitations and Map Inaccuracies . .	68
3.3.3	Human Trial Experiments with the QPP	73
3.4	Chapter Conclusions	79
4	A Multimodal Approach to Qualitative Navigation using Speech and Sketch	82
4.1	Multimodal Strategy for Communicating Navigation Instructions	83
4.1.1	System Architecture	85
4.1.2	Natural Language Processing	88
4.1.3	Dataset	92
4.2	Improving Sketch Recognition using Speech	97
4.2.1	Probabilistic Sketch Representation	98
4.2.2	Multimodal Recognition Results	106
4.3	Evolutionary Approach to Simultaneous Sketch Recognition and Map Alignment	112
4.3.1	EAMMA: An Evolutionary Algorithm for Multimodal Map Association	113
4.3.2	Experiments and Results	122
4.4	Speech Augmented Qualitative Path Planner (QPP*)	134
4.4.1	Overview of the Qualitative Path Planner	134
4.4.2	Augmented Cost Function Using Speech Cues	137
4.4.3	Results and Comparison Between QPP and QPP*	140
4.5	Chapter Conclusions	147
5	Summary	150
A	Appendix	153
	References	157

LIST OF TABLES

2.1	Explanation of NASA-Task Load Index sub-scales. Each scale was rated from 1–100.	39
3.1	Mean (Standard Error) statistics from the human trials.	76
A.1	Features used for interstroke inference. Distances are measured in pixels.	153
A.2	Features used for gesture and stroke inference. (Features marked with an * are used for gestures only.) Distances are measured in pixels.	154
A.3	Verbal observation features o_v . Distance is measured in pixels, time measured in seconds.	156

LIST OF FIGURES

2.1	An operator using the sketch interface to control a mobile robot.	10
2.2	Example of a two-gesture stroke modeled using a variable duration hidden Markov model.	14
2.3	The set of gesture classes \bar{G} . Observation features o_{g_k} are extracted from pixel-level data, and are listed in Table A.2 in Appendix A.	18
2.4	Results of classification on 1021 test gestures. Rows correspond to the true class c_g , columns correspond to the recognized class. .	20
2.5	Results of classification on 1398 test strokes. Rows correspond to the true class c_s , columns correspond to the recognized class. . .	21
2.6	Results of classification on 1304 test interstrokes. Rows correspond to the true class c_i , columns correspond to the recognized class.	23
2.7	An example sketch with seven gestures and eleven strokes (each stroke is drawn in a different color for clarity).	24
2.8	(Left) An example sketch (a single <code>arrow</code> gesture) composed of two strokes. (Right) The generated search tree.	25
2.9	Effect of stroke number $N^{[n]}$ and α on the modified likelihood \mathcal{L}'_n . The value of α determines the degree to which nodes are ‘penalized’ in the search.	27
2.10	Algorithm accuracy and efficiency for 87 test sketches of 5–8 strokes for $M = 9$	28
2.11	Robot used during experiments equipped with an IMU, 180° SICK laser range finder, and three cameras with a combined 150° field of view.	30
2.12	GUI panel for answering inequality problems.	31
2.13	Dual-monitor user interface for human-operator experiments. The left monitor displays an overhead view of the robot, obstacles in the environment, and mapped POIs. The right monitor streams the 150° camera field of view.	31
2.14	The 23m × 18m experimental environment, which was not visible to the operators from the command station. Two POI configurations were used to prevent operators from easily learning the map.	33

2.15	Sequence of events for an example sketch. The robot is displayed as a rectangle with a line pointing out the front (a selected robot also displays its 150° 3-camera field of view). Obstacles observed with the 180° laser range finder are displayed as wire polygons on the map. In the top left panel, the user sketches three gestures: <code>circle</code> , <code>path</code> , and <code>triangle</code> . After the user presses the ‘Send Sketch’ button, the recognized sketch with the highest likelihood is rendered in orange, superimposed on the original sketch (shown in top right panel). Upon confirmation from the user, the robot in the <code>circle</code> is selected and follows the commanded path, and the triangle POI’s location is placed on the map (bottom panels).	35
2.16	Operator performance for primary mission objectives.	37
2.17	Operator performance for secondary mission objective and subjective workload measures.	38
2.18	Post-experiment survey responses. 10 out of 16 users preferred Sketch mode over Waypoint mode.	40
3.1	(top) Human-provided map of the environment with desired path drawn in blue. (bottom) True map of the environment, which the robot does not have access to. True landmark locations are shown as red triangles.	47
3.2	A block diagram of the system architecture. A robot receives commands from a human user in the form of an approximate map of landmarks and a desired path. The QPP solves for waypoints in the estimated environment that best maintains appropriate spatial relationships to the observed landmarks.	48
3.3	For each landmark, the spring constant k_i^j is calculated as a function of the distance between waypoint w_i^S and landmark ℓ_j^S , and the uncertainty in the estimated location of the landmark Σ_j^E . . .	51
3.4	Waypoints (*) are extracted at points along the path that intersect with lines in the Voronoi-Delaunay graph.	54
3.5	A sketch map consisting of landmarks (triangles) and waypoints (*). The desired path passes between five landmark pairs, or ‘gates’.	56
3.6	The planned paths obtained from the QPP using: (a) sparse waypoints defined by user, (b) the path with waypoint augmentation using the Voronoi-Delaunay graph, and (c) a super-segmented path.	56
3.7	Three sensor fields of view shown as a function of the landmark effective distance (ED) in a randomly-generated environment of 20 landmarks. In this example, the effective distance is 24.7m. . .	62

3.8	The modified QPP is able to plan around an obstacle by incorporating an elliptical constraint (Equation 3.9) in the waypoint optimization.	64
3.9	The sketched map (left) includes eight landmarks, with a path passing between four pairs in a straight line. In the true environment (right), the landmark locations require the robot trajectory to weave between landmarks pairs in order to maintain appropriate spatial relationships. (sensor range = 46m or 1.5ED)	66
3.10	The sketched map (left) shows a path that passes around and between nine landmarks. In the true environment (right), three landmarks are not present, due to their not existing or not being sensed by the robot. (sensor range = 78m or 1.5ED)	67
3.11	The environment used for the sensitivity analysis. The sensor radius was varied $r \in [0.3, 0.75, 1.2ED]$, and Gaussian noise was added to each landmark location $\sim \mathcal{N}(0, \sigma^2)$ where $\sigma \in [0.06, 0.12, 0.18, 0.24, 0.3ED]$	68
3.12	Results of a sensitivity study for the map in Figure 3.5. Each data point represents the mean and standard error of 50 Monte Carlo simulations. The number of successful landmark gates is plotted as a function of the landmark distortions σ for (a) small, (b) medium, and (c) large sensor ranges, r	70
3.13	The QPP trajectory for the same sketch and true environment ($\sigma = 6m$, or 0.18ED) for three different sensor radii.	70
3.14	The interface used for collecting sketch maps in human trials. On the left, an image is shown of the environment consisting of a robot, twelve landmarks (numbered cones), a goal position (X), and a desired path (for sets <code>PathAny</code> and <code>PathAll</code> only). On the right, the user sketches a map to illustrate how the robot should navigate to the goal.	72
3.15	Three users' sketch maps for the <code>PathAny</code> set (Map 4, see Figure 3.14(left)). Each user indicates only the landmarks s/he thinks are necessary for navigation, and draws a path to the goal.	72
3.16	Trajectories generated by QPP in the true environment based on sketched maps provided by ten human users for $r = 1.5ED$	75
3.17	Average final goal offset as a function of sensor range r for the case where all landmarks were included in the sketch map (<code>FreeAll</code> and <code>PathAll</code>) and the case where the user chose which landmarks to sketch (<code>FreeAny</code> and <code>PathAny</code>).	77
4.1	A block diagram of the system architecture. Speech and sketch inputs from a human operator are used to interpret and execute qualitative navigation instructions.	86

4.2	Graphical representation of natural language processing. For each verbal phrase v spoken, Speech SDK provides several hypothesis phrases v_h consisting of one or more words. Words that belong to the list of pre-defined grammar utterances $u_{h,j}$ are extracted and used for sketch recognition, map alignment, and qualitative navigation.	90
4.3	Users were given overhead images of six scenes consisting of various objects and a desired route and asked to provide navigation instructions through the scene using speech and sketch. . . .	93
4.4	For Phases 3–4, users were instructed to sketch objects in the map as shown. Notice that the square and circle gestures are used for multiple different objects.	93
4.5	For Phases 4–5, users were instructed to sketch objects in the map as shown. Each gesture corresponds to a distinct object type. . .	95
4.6	Two users' speech and sketch inputs for the same scene in Phase 1.	96
4.7	Graphical representation of an example speech and sketch input, where the sketch consists of five strokes $s_{j,j} \in [1, \dots, 5]$ and two gestures g_j . The user's speech v produces three hypothesis phrases v_j	100
4.8	Classification accuracy for both gestures and strokes significantly improve by using multimodal observation features rather than speech or sketch alone.	107
4.9	Gesture classification accuracy in Phase 3 using sketch features alone, due to the ambiguous nature of the gestures. Four object classes were sketched as squares, and two classes were sketched as circles. Speech observation features help to disambiguate these gestures, significantly increasing classification accuracy. . .	108
4.10	The genetic representation of an example individual, which encodes associations between strokes in the sketched map and objects in the observed environment.	116
4.11	The five types of reproduction used by EAMMA to evolve the population.	120
4.12	The number of generations evolved by EAMMA before convergence. Convergence was triggered when the best individual remained unchanged for 100 consecutive generations.	123
4.13	Sketch recognition and map alignment results using EAMMA with different fitness functions.	124
4.14	For treatments that penalize for unassociated objects in the environment (A and E), EAMMA performs worse when the sketch map does not include all objects in the environment. This emphasizes the importance of choosing an appropriate fitness function.	127

4.15	The performance of EAMMA depends on the quality and type of sketch and speech input, and is therefore not uniform across different phases and subjects.	128
4.16	Some users (such as Subject 2) provided sketch maps with very few strokes, improving map association accuracy. Others (such as Subject 3) used many strokes, making it more difficult for EAMMA to correctly associate the sketch map.	130
4.17	EAMMA converges to solutions of higher fitness values (as defined by Treatment A) than a standard hillclimber and simulated annealing, resulting in higher map association and stroke recognition accuracies.	132
4.18	EAMMA results for Treatments A and D. Plots (a) and (c) illustrate that errors in sketch recognition can be reduced by associating objects in the sketch map to those observed in the environment.	133
4.19	The location of waypoint w_i^E is influenced by surrounding landmarks ℓ_j^E via virtual springs k_i^j . The original QPP algorithm defines k_i^j as a function of the distance between waypoint w_i^S and landmark ℓ_j^S , and the uncertainty in the estimated location of the landmark Σ_j^E	135
4.20	An example sketch where the path passes between two landmarks (colored for clarity). Using the original QPP algorithm, even ‘non-critical’ landmarks (unshaded objects) will influence the optimized trajectory. QPP* augments each landmarks’ influence along the path according to referenced objects in the user’s speech.	138
4.21	Typical examples of the trajectories planned by QPP and QPP*. In most cases where no data association errors are made between the sketch map and objects in the environment, both algorithms perform similarly.	141
4.22	An error in data association (the sketched chair is associated with the toolbox in the environment) causes QPP to fail to plan a reasonable trajectory. QPP* is able to compensate for this error by because the user makes no mention of a “toolbox” while drawing the path, so the wrongly associated landmark is virtually ‘ignored’.	142
4.23	Due to an error in data association, QPP fails to plan a reasonable trajectory. QPP* is able to compensate for this error by because the user makes no mention of a “trash can” while drawing the path, so the wrongly associated landmark is virtually ‘ignored’.	144
4.24	Performance of QPP and QPP* with the addition of simulated ‘non-critical’ landmarks in the sketch map.	146

CHAPTER 1

INTRODUCTION

One of the current challenges facing the robotics community is the need for intelligent interfaces to facilitate more natural human-robot interaction. Especially in search-and-rescue and military applications, it is increasingly necessary that humans be able to communicate naturally and efficiently with teams of collaborating agents, be they humans, robots, or both. In particular, the command of teams consisting of both humans and robots should require a single communication strategy, so the operator need not be concerned with the type or capacity of each individual agent. This motivates the development of an interface that allows humans to communicate with robots in the same flexible and natural way as they would communicate with other humans. This thesis presents several methods for conveying navigation instructions to a mobile robot using sketch and speech inputs, with the aim of allowing users to communicate naturally with a semi-autonomous robot while placing as few restrictions on the user as possible.

1.1 Natural Robot Interface via Free Sketch Recognition

In order to facilitate natural human-robot interaction, this thesis first proposes an interface to allow operators to command robotic agents by drawing free-hand sketches. Efficient and accurate recognition of these sketched commands are vital to the efficacy of such a system, and it should remain flexible and robust to different sketching styles. Much work has been done in the area of handwriting and sketch recognition in the past few decades, including recognition of single-stroke characters [6, 7], pre-segmented multi-stroke gestures, ambigu-

ous transition multi-stroke gestures [8, 9, 10], and even interspersed sketches [11, 12]. Several models have been successfully applied to sketch recognition in literature, including HMMs [13, 14], neural networks [15, 16], and sometimes both [17]. In this work, the ambiguous transition multi-stroke gesture problem is considered, where segmentation and recognition are performed simultaneously using a variable duration hidden Markov model (VDHMM).

There has been some recent work in commanding robots using hand-drawn sketches, with the focus primarily on qualitative mapping [18, 19, 20] and navigation problems [21, 22, 23, 24], where sketch recognition is hard-coded or nearly-trivial. For example in [22], an “object” is defined as a closed polygon if the Euclidian distance between the starting and end points fall within an empirically set threshold, whereas a “path” is interpreted as any line segment that is not recognized as a closed polygon and has a total length greater than a set threshold. The authors do not address the effect of incorrect interpretations of the sketches resulting from variations in sketching styles. In contrast, the work presented here defines sketches within a robust probabilistic framework based on a learned model using local and global features. This enables the use of flexible gestures without the need for hard-coding or template-matching, and users are not restricted to drawing gestures in a particular way.

A sketch interface was successfully used in [6] and [25] to dictate formation commands to a small group of robots using a PDA (Personal Digital Assistant). The authors implemented a hidden Markov model (HMM) symbol recognition component to identify and distinguish between a small set of simple sketched commands. While they also approached sketch recognition as a probabilistic classification problem, users were constrained to using only rigidly defined

single-stroke gestures. This is an obvious limitation of their interface, which does not allow the user to sketch truly naturally. Additionally, their high false alarm rate was reduced during a post-processing phase where gestures were separated into pre-defined “open” and “closed” categories.

Chapter 2 presents an approach for commanding mobile robots using a probabilistic multi-stroke sketch interface. Drawing from prior work in handwriting recognition, sketches are modeled as a variable duration hidden Markov model, where the distributions on the states and transitions are learned from training data. A forward search algorithm is used to find the most likely sketch given the observations on the strokes, interstrokes, and gestures. A heuristic is implemented to discourage breadth-first search behavior, and is shown to greatly reduce computation time while sacrificing little accuracy. To avoid recognition errors, the recognized sketch is displayed to the user for confirmation; a rejection prompts the algorithm to search for and display the next most likely sketch. Upon confirmation of the recognized sketch, the robot executes the appropriate behaviors. A set of experiments was conducted in which operators controlled a single mobile robot in an indoor search-and-identify mission. Operators performed two missions using the proposed sketch interface and two missions using a more conventional point-and-click interface. On average, missions conducted using sketch control were performed as well or better than those using the point-and-click interface, and results from user surveys indicate that more operators preferred using sketch control.

1.2 Qualitative Navigation Strategies

In recent years, significant effort has been applied towards developing mobile robots to perform many tasks that were once undertaken solely by humans. In hazardous or inaccessible environments, such as those encountered in disaster-relief operations or planetary exploration, mobile robots are often able to navigate areas that humans cannot [26]. In the future, autonomous urban vehicles and domestic assistance robots will offer increased convenience and independence to their human users by performing navigation tasks semi- or completely autonomously [27].

As mobile robots become more ubiquitous, communication between such agents and their human counterparts should be intuitive and robust, emulating the way humans interact with each other. As an example, humans often communicate basic navigation tasks to each other using approximate spatial relationships to observable landmarks [28], without requiring a precise map (for example, “walk past the computers and take a left at the elevator”). This type of human-robot communication would be particularly useful when a true map is unavailable or the environment is changing with time. A robot operating around an earthquake site may not be able to navigate using a city map in global coordinates, and should instead plan its path with respect to unique observable landmarks such as cars, trees, and buildings. Interpreting such navigation instructions by computers has been an active area of research in recent years [20, 80, 81, 82]. On one hand, it is desirable to allow the human input to be as natural and flexible as possible. At the same time, it is necessary that efficient and reliable intention recognition be achieved. These competing objectives make it difficult to develop a robust system for qualitative robot navigation.

This thesis proposes an approach for controlling a mobile robot using a qualitative map consisting of landmarks and path waypoints, such as one sketched by a human. The sketch need not contain precise information about the entire environment, but only information relevant to the navigation task. Additionally, the sketched map may not be quantitatively accurate, but should be *qualitatively* similar to the true environment.

The challenge of robot navigation using qualitative maps has been addressed in recent years. Navigation using topological diagrams of structured environments has been presented for which schematic maps are used to encode qualitative spatial information about the physical environment [29, 30, 31, 32, 33]. Navigation is accomplished by performing spatial reasoning and matching sensory data with modeled sensor behavior at intermediate goal points along the route. These approaches rely on structured environments (such as an office building) and are not robust to large map inaccuracies.

In [34], an operator works from a rough initial map indicating the approximate current location of the robot and a path of ‘via points’ to a goal. The relations to expected visible landmarks (encoded in the Landmark Egosphere, LES) and sensed landmarks (encoded in the Sensory Egosphere, SES) are compared. A via point is considered reached if the LES matches the SES to within some pre-defined tolerance. Similarly, in [24, 22] and [35], a sketch containing approximate landmarks and a path is drawn on a PDA by a human operator. At ‘crucial nodes’ along the sketched route, spatial relations between the robot and surrounding objects define Qualitative Landmark States (QLS) and associated robot commands. As the robot navigates the true environment, the observed QLSs are identified and matched with those extracted from the sketched route,

and the corresponding pre-calculated commands are executed. This approach can account for some degree of uncertainty and inconsistencies in the sketch subject to appropriate tuning and pre-defined thresholds, but is generally limited to straight-line paths, is sensitive to missing landmarks, and is not able to recover if the robot misses a QLS or strays too far from the desired route.

In Chapter 3, a method for controlling a mobile robot using qualitative inputs in the context of an approximate map, such as one sketched by a human, is presented. By defining a desired trajectory with respect to observable landmarks, human operators can send semi-autonomous robots into areas for which a true map is not available. Waypoint planning is formulated as a quadratic optimization problem which takes advantage of the probabilistic representation of the observed environment and the uncertain human input, resulting in robot trajectories in the true environment that are qualitatively similar to those provided by the human. This chapter formally presents a methodology in which waypoints are extracted from a hand-drawn sketch, and obstacle avoidance is naturally accommodated through the addition of constraints in the optimization problem. A sensitivity analysis is performed to study how map distortions, sensor constraints, and *a priori* knowledge of the map orientation affect the performance of the planner. Lastly, a set of human experiments is presented to demonstrate the robustness of the planner to different users' sketched maps and to illustrate the efficacy of such a method for mobile robot control.

1.3 Multimodal Communication for Robot Control

In order to support more flexible, efficient and expressive means of human-computer interaction, multimodal interfaces have become popular in recent decades. Multimodal interfaces allow the user to communicate using multiple different methods, often simultaneously. Since single modes of communication (such as speech, gesturing, and drawing) are not generally effective across all tasks and environments, this provides the user freedom to use a combination of modalities, or to switch to a different modality better-suited for the particular task at hand. For example, speech is well-suited for providing descriptive information such as identifying objects and describing physical features, while pen inputs are effective at expressing spatial relations [34, 24] and illustrating graphical structures such as circuit diagrams and family trees [11, 36].

Multimodal interfaces have been applied to a wide variety of tasks such as the design of mechanical devices [37], unmanned vehicle control [38, 39], and city-guide information retrieval [40]. Such interfaces commonly fuse input from speech and physical gestures (such as pointing) [41, 42, 43] or speech and pen gestures [44, 45, 46]. One of the primary advantages of multimodal design is its ability to more effectively prevent and recover from errors. Error *avoidance* is achieved primarily at the user-level, because the user is able to choose the most appropriate method of communication for each task. Additionally, users often use simpler language when additional communication modes are available, simplifying the natural language processing task [47]. Error *recovery* is accomplished due to mutual disambiguation of input signals from different modes. Since multiple modes are generally used simultaneously to express duplicate or complementary information, the input provided via one mode can often par-

tially or wholly resolve ambiguity of information provided via another input mode. Fusing two or more information sources can therefore be an effective means of reducing recognition uncertainty and increasing robustness.

Multimodal interfaces using spoken and pen-based input are particularly powerful for tasks such as describing objects or spatial layouts because these inputs provide complementary capabilities. Compared to speech-only interaction, it has been demonstrated that multimodal pen/voice interaction during visual-spatial tasks can result in 10% faster task completion time, 36% fewer task-critical content errors, 50% fewer spontaneous disfluencies, and also shorter and more simplified linguistic constructions [47]. Users were shown to prefer using pen input over speech to effectively convey precise locations, areas, and other spatial information.

Chapter 4 presents a multimodal interface for communicating navigation instructions using a combination of speech and sketch input. Motivated by how humans communicate with one another, an approximate sketched map and verbal route instructions are provided by a human user and used to perform qualitative navigation. Users are able to speak truly naturally, and verbal inputs are not required to conform to pre-defined grammar structures. Drawing on previous work in sketch recognition and natural language processing, observations from pixel-level data and spoken utterances are used to perform sketch recognition. Recognition accuracy is shown to be significantly improved in the multimodal case compared to either mode individually. Second, an evolutionary algorithm is proposed for performing data association between objects in the sketch map and those in the robot's observed environment. This approach solves the multiobjective optimization problem of maximizing the

sketch segmentation and recognition likelihood, minimizing object location errors, and maximizing the number of associated objects in the sketched map. The algorithm is shown to not only successfully perform map association, but also improves sketch recognition accuracy. Lastly, an extension to the previously proposed Qualitative Path Planner is presented, which plans a trajectory through the observed environment using spatial information extracted from the user's sketch and augmenting the influence of landmarks according to the user's speech. Each algorithm is evaluated using a dataset collected from 12 users. The final result is a multimodal interface that allows human users to naturally and effectively communicate navigation instructions to a robotic agent.

1.4 Contributions

The major contributions presented in this dissertation are:

- A fully probabilistic approach to multi-stroke sketch recognition using machine learning techniques to increase recognition accuracy and flexibility to the user.
- A method for controlling semi-autonomous mobile robot navigation in the absence of a metrical map using qualitative inputs provided by a human sketch.
- A multimodal interface for communicating route instructions by fusing sketch maps and verbal inputs without restricting the human's speech or performing high-level language parsing.

CHAPTER 2

A SKETCH INTERFACE FOR ROBUST AND NATURAL ROBOT CONTROL

This chapter presents a sketch control interface to facilitate effective and natural human-robot interaction for a set of supervisory tasks. The interface, shown in Figure 2.1, allows an operator to control a robot remotely by inputting a freely-drawn sketch using a mouse or pen tablet. Similar in nature to a football playbook, sketched symbols (called ‘gestures’) represent particular actions or behaviors to be executed by the mobile agent(s). The sketch recognition algorithm, which learns probabilistic models of the multi-stroke sketch components from training data, can be trained on multiple users for increased robustness or on a single user for increased accuracy if the intended operator is known. This method enables users to sketch naturally, as if they were sketching for another person. Recognition is performed on-line, and executed upon confirmation from the user.

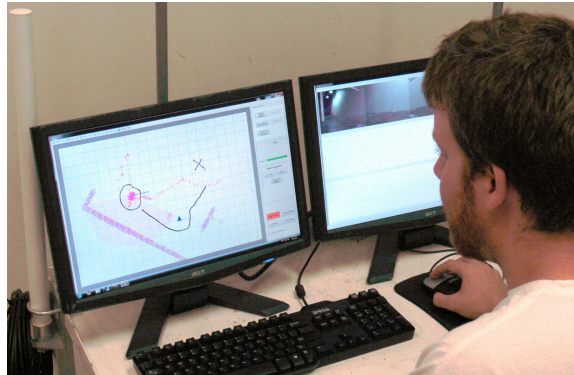


Figure 2.1: An operator using the sketch interface to control a mobile robot.

Drawing from prior work in handwriting recognition, the proposed method offers several advantages over previous approaches discussed in 1.1. First, by learning sketch models from training data, rigid *a priori* assumptions about how gestures should be drawn, such as defining an “X” as two intersecting lines of a particular length, are largely avoided. The model can be learned on many people, providing a more robust interface for a wide range of users; it can also be learned on a single user to increase recognition accuracy if the intended operator is known. This helps to shift the burden of correct sketch recognition from the user to the machine, allowing the user to sketch more naturally. Second, a probabilistic framework enables an intelligent search of the space of possible sketch interpretations, the size of which grows exponentially with the number of strokes. If the ‘best’ solution (i.e., the sketch interpretation with the highest likelihood) is rejected by the user, the interface proposes the ‘next best’ solution. This is both more convenient to the user than re-drawing the sketch from scratch, and also useful for updating the model on-line. Lastly, by framing the sketch recognition problem as a variable duration hidden Markov model, the proposed method supports flexible and multi-stroke gestures, and can be extended to incorporate multi-modal communication such as verbal cues.

Some initial experiments using the proposed sketch interface were presented in [48], in which operators performed a short search-and-identify mission using a pen tablet to send robot commands. In this work, a similar experiment using the sketch recognition framework and interface from [48] is conducted in a larger, more complex environment. 16 operators performed four missions each, two using the proposed sketch control interface and two using a more conventional point-and-click interface with buttons and menus. A secondary mission objective was included to examine the operators’ ability to multi-task and main-

tain situation awareness. Both interfaces are evaluated using objective measures corresponding to how well the mission objectives were met, as well as subjective measures such as workload and operator preference.

This chapter is organized as follows. Section 2.1 outlines an approach for using a variable duration hidden Markov model to define a sketch as a combination of gestures, strokes, and interstrokes. The distributions of the Markov model are learned using multinomial logistic regression, details of which are presented in Section 2.2. Section 2.3 describes the sketch recognition algorithm and introduces a heuristic for improving algorithm efficiency. A set of experiments is presented in Section 2.4, in which users control a robot using two different interfaces to perform an indoor search-and-identify mission. Experimental results are presented in Section 2.5, followed by discussion and conclusions in Section 2.6.

2.1 Probabilistic Sketch Definition

In this work, a sketch is characterized as a sequence of strokes, gestures, and interstrokes, defined as follows:

- **Stroke:** A stroke s_k is a collection of pixels generated during a single pen-down (or mouse-down) instance. Each stroke corresponds to a particular stroke class $s_k = c_s \in \bar{S}$, where the set of M_s possible stroke classes \bar{S} is finite and known. In the current implementation, pixel data is recorded every 15ms.
- **Gesture:** A gesture g_k is a complete shape or symbol composed of d_k strokes, where $d_k \in \mathbb{N}_1$ is the (unobserved) gesture duration. Each ges-

ture corresponds to a particular gesture class $g_k = c_g \in \bar{G}$, where the set of M_g possible gesture classes \bar{G} is finite and known.

- **Interstroke:** An interstroke i_k is the transition between two consecutive strokes. The corresponding interstroke class $i_k = c_i \in \bar{I}$ defines whether a stroke belongs to the same gesture as the previous stroke (i.e., a self-transition on the gesture) or is the start of a new gesture.
- **Sketch:** A sketch is a series of N_S strokes and N_G gestures $G = [g_1, \dots, g_{N_G}]$, where the total number of gestures drawn N_G is unknown (because d_k is unobserved).

Here, it is assumed that each stroke belongs only one gesture and that strokes are not interspersed, i.e., users do not return to a gesture after starting a new one. The total number of strokes in a complete sketch $N_S = \sum_{k=1}^{N_G} d_k$ is known, and stroke start/end points are identified by mouse down/up instances. The stroke classes in \bar{S} coincide with the gesture classes in \bar{G} (therefore $M_s = M_g = M$); for example, if g_k is a `triangle` gesture, the strokes composing that gesture S_k are `triangle` strokes. This is similar to existing methods that categorize strokes as primitives, such as lines or arcs [11]. The key difference here is that the primitives are not defined *a priori*, rather the stroke model is *learned* such that discriminating features are used for inference (e.g., it learns what makes `triangle` strokes fundamentally different than `path` strokes).

This work draws from prior work in handwriting recognition by implementing a variation of the variable duration hidden Markov model (VDHMM, sometimes also referred to as hidden semi-Markov model). VDHMMs have been studied and successfully implemented for recognizing handwritten text, specifically in cases for which the segmentation problem must be solved, such as with

cursive and Arabic handwriting [49, 50, 51, 52, 53, 54]. In a VDHMM, the system is assumed to stay in a particular state for some (generally unknown and non-constant) duration before transitioning to the next state. The state transition probabilities are then conditional on both the current state and its duration.

Figure 2.2 illustrates an example of a VDHMM modeling a sketch of $N_S = 5$ strokes and $N_G = 2$ gestures, where shaded nodes represent observations, unshaded nodes represent hidden (unobserved) variables, and arrows represent conditional dependencies. In this example, the first gesture g_1 has a duration of $d_1 = 3$ strokes, and the second gesture g_2 has a duration of $d_2 = 2$ strokes. Inter-stroke nodes i_1, i_2 , and i_4 represent self-transitions on the gestures, while node i_3 represents a transition from gesture g_1 to gesture g_2 . The goal of the sketch recognition algorithm is to identify the specific sequence of strokes, interstrokes and gestures sketched, given observations extracted from pixel-level data. This is accomplished by maximizing the observation likelihood:

$$p(\text{sketch}|O) = p(G, S, I|O) \quad (2.1)$$

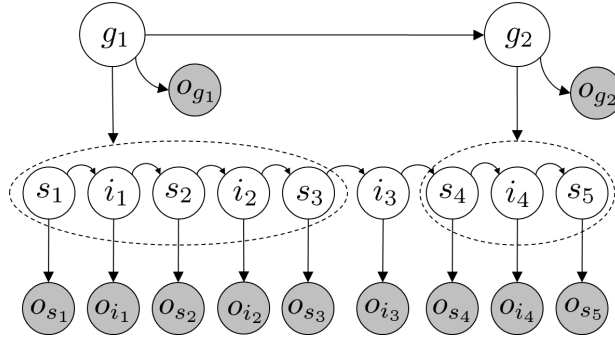


Figure 2.2: Example of a two-gesture stroke modeled using a variable duration hidden Markov model.

Here, $O = \{O_G, O_S, O_I\}$ are the observations on the gestures, strokes, and inter-strokes respectively. Equation 2.1 can be factorized as:

$$p(G, S, I|O) = p(S, I|O_S, O_I) p(G|S, I, O_G) \quad (2.2)$$

The observations O_G, O_S, O_I are features extracted from the pixel data, to be described in Section 2.2.1. Assuming a first-order Markov model for strokes and interstrokes, and uniform marginal distributions for all gesture, stroke, and interstroke classes, the terms on the right-hand side of Equation 2.2 can be written as:

$$p(S, I|O_S, O_I) \propto p(s_1|o_{s_1}) \prod_{k=1}^{N_S-1} p(s_{k+1}|i_k) p(s_{k+1}|o_{s_{k+1}}) p(i_k|s_k) p(i_k|o_{i_k}) \quad (2.3)$$

$$p(G|S, I, O_G) = \prod_{k=1}^{N_G} p(g_k|S_k, I_k, o_{g_k}, g_{k-1}) \propto \prod_{k=1}^{N_G} p(g_k|S_k, I_k) p(g_k|o_{g_k}) p(g_k|g_{k-1}) \quad (2.4)$$

where S_k and I_k are the strokes and interstrokes that compose gesture g_k . The distributions in Equations 2.3 and 2.4 are learned from training data, allowing for the inclusion of a variety of scripts and icons without the need to hard-code their attributes. This flexibility is key in enabling an arbitrary selection of intuitive and natural gestures for the basic units in the lexicon of robot commands.

2.2 Learning the Variable Duration Hidden Markov Model

In this work, a few assumptions are made before training the HMM to reduce the number of distributions in Equations 2.3 and 2.4 that must be learned and the amount of training data required. First, the probability distribution of transitioning from one gesture class to another is assumed to be uniform (g_k is inde-

pendent of g_{k-1}). This assumption is not necessary (and may be inappropriate for some applications), but it provides a reasonable balance between complexity and accuracy for the application presented here. Second, gesture ‘mistakes’ or otherwise unknown gestures are not considered, although this could also be easily incorporated into the framework by including an $(M + 1)$ th other gesture class. Third, gesture g_k must be of the same class as its associated strokes S_k (e.g., a `triangle` can only be drawn with `triangle` strokes). Lastly, while the number of gestures N_G in the sketch is unknown, the total number of strokes N_S is known (i.e., the beginning and end of each full sketch is unambiguous).

Some parts of the distribution in Equation 2.3 are known *a priori*. For example, $p(s_{k+1}|i_k) = 0$ when i_k is a gesture self-transition and the class of the following stroke s_{k+1} does not match (e.g., i_k is a `box` self-transition and s_{k+1} is not a `box` stroke). Similarly, $p(i_k|s_k) = 0$ when i_k is a gesture self-transition and the class of the preceding stroke s_k does not match. The remaining portions of Equations 2.3 and 2.4 must still be defined: $p(i_k|o_{i_k})$, $p(s_k|o_{s_k})$, and $p(g_k|o_{g_k})$. Each of these conditional probabilities is treated as a multinomial classification problem, the distributions of which are determined using a supervised learning algorithm, Sparse Multinomial Logistic Regression (SMLR) [55]. SMLR is a true multiclass formulation based on multinomial logistic regression [56], i.e., it doesn’t reduce the full multiclass problem into multiple binary classification problems. Instead, it learns weight vectors \mathbf{w} such that the likelihood of y being classified as class c for a set of observed features \mathbf{x} is given by:

$$\mathcal{L}_c(y) \equiv p(y = c|\mathbf{x}; \mathbf{w}) = \frac{\exp(\mathbf{w}_c^T \mathbf{x})}{\sum_{j=1}^m \exp(\mathbf{w}_j^T \mathbf{x})} \quad (2.5)$$

where m is the total number of possible classes. In this work, y corresponds to g_k , s_k , or i_k and \mathbf{x} corresponds to observations o_{g_k} , o_{s_k} or o_{i_k} , respectively.

SMLR incorporates sparsity-promoting (Laplacian) priors to encourage the components in the weight vectors to be either significantly large or exactly zero. Although calculating the maximum *a posteriori* multinomial regression with a Laplacian prior scales unfavorably with the number of bases, the component-wise update equation has a monotonically increasing closed form solution and results in a computation cost and storage requirement linear in the number of bases [55]. As with relevance vector machines (RVMs) [57], SMLR uses Bayesian inference to provide probabilistic classification $\mathcal{L}_c(y)$ (i.e., there is a notion of *how* certain a set of features belongs to a particular class c) and can incorporate arbitrary bases, including non-Mercer kernels. SMLR, however, converges to a unique maximum and is not at risk of converging to local minima as RVMs are [58].

Once the weight vectors w are learned from training data, the likelihood that a new example y corresponds to class c is straightforwardly calculated using Equation 4.5. A separate set of weight vectors is learned for each observation likelihood from Equations 2.3 and 2.4: $p(i_k|o_{i_k})$, $p(s_k|o_{s_k})$, and $p(g_k|o_{g_k})$. SMLR provides a convenient and intuitive way to learn these distributions, however it does require a sufficiently large training data set. Section 2.2.1 describes the observation features o_{g_k} , o_{s_k} and o_{i_k} used in this work.

2.2.1 Observation Feature Set

To learn the distributions in Equations 2.3 and 2.4, data was collected from fourteen users for $M = 9$ gesture classes, shown in Figure 2.3(a). Users were told to draw the gestures “naturally” and were given minimal instructions regarding

how gestures should look. For each gesture drawn, 94 features (corresponding to o_{g_k}) were extracted from the pixel data. Some of these features were adopted from a portion of the g-48 set presented in [59] and [60], which was developed to be generally well-suited for identifying multiple-stroke gestures. Also included in o_{g_k} are features corresponding to initial orientation angles, the amount of time spent drawing each gesture, the total number of strokes, and clockwise/counter-clockwise orientation.

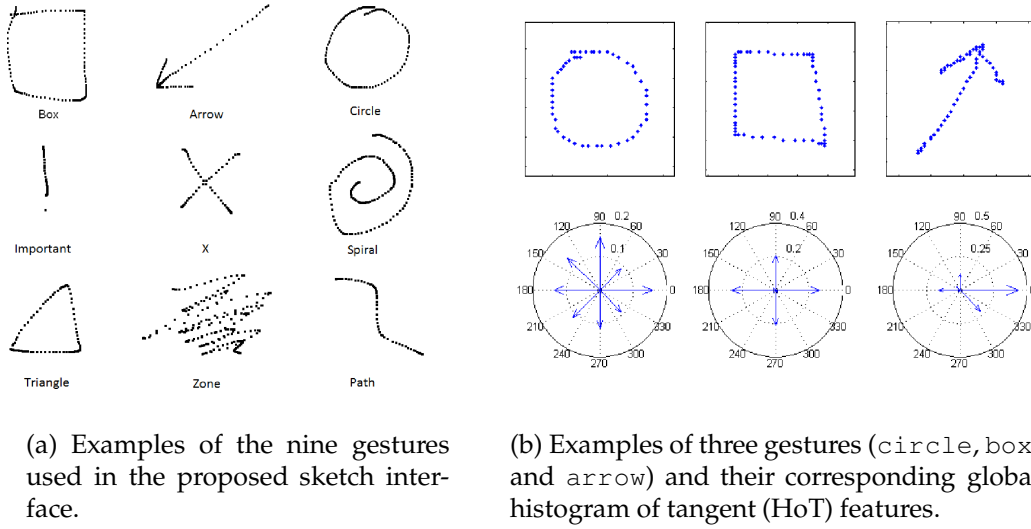


Figure 2.3: The set of gesture classes \bar{G} . Observation features o_{g_k} are extracted from pixel-level data, and are listed in Table A.2 in Appendix A.

An additional set of features, referred to as Histogram of Tangents (HoT features), is proposed in this work. These features are defined by calculating the stroke tangent angle between each pair of neighboring pixels. The angles are discretized into eight bins, normalized to 1, and ‘shifted’ such that the first bin is aligned with either the dominant angle (defining eight global HoT features) or the initial angle (defining another eight global HoT features). A total of 16 global and 32 local (piecewise) HoT features are calculated for each ges-

ture. Figure 2.3(b) plots the set of global HoT features for three gestures with the dominant angle aligned with 0° , and illustrates how these features might be useful for distinguishing between different gesture classes. For the `circle` gesture, the distribution of HoT features is relatively uniform. However for the `box` gesture, the HoT features at $0^\circ, 90^\circ, 180^\circ, 270^\circ$ dominate, representing the four straight edges of the `box`. The HoT features extracted from the `arrow` gesture reveal one dominant angle and two or three less dominant non-zero angles. The addition of these features was shown to increase average recognition accuracy an additional 6% over the g-48 features alone.

91 features were extracted as stroke observations o_{s_k} , and were the same as those used for gestures (minus three redundant features corresponding to number of strokes, average time drawing each stroke, and total time drawing the gesture). Nine interstroke features (o_{i_k}) were defined by extracting information from consecutive strokes, including relative positions, sizes, and orientations, as well as temporal information. Refer to Tables A.1 and A.2 in the Appendix for the detailed list of observation features used in this work.

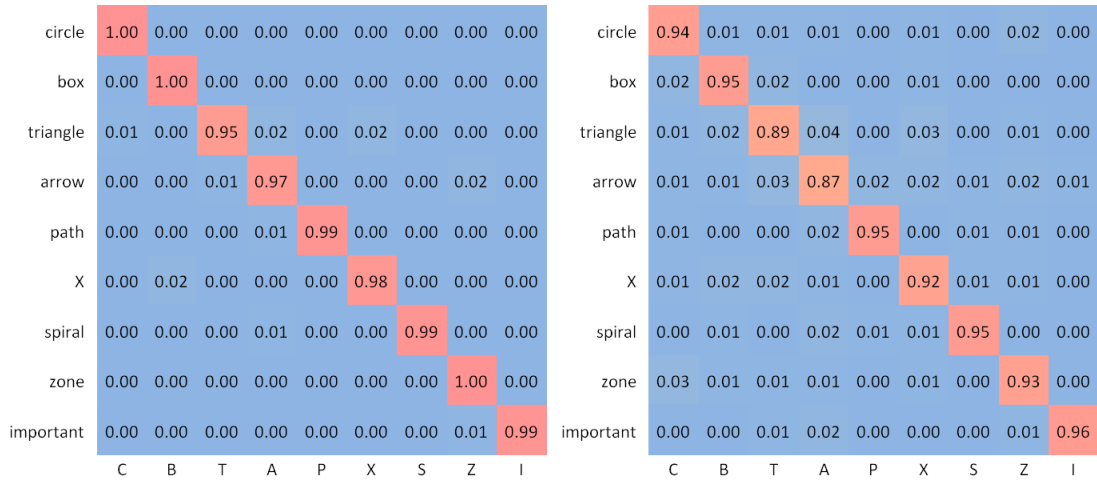
2.2.2 Gesture Classification

The gesture classifier, which returns $p(g_k = c_g | o_{g_k})$, was learned using SMLR on 720 gestures (80 of each class $c_g \in \bar{G}$). Figure 2.4(a) shows the resulting confusion matrix for 1021 test gestures, which were classified with 98.6% accuracy (comparable to previously published results in handwriting and sketch recognition). Figure 2.4(b) presents the average likelihoods calculated by the learned regression model from Equation 4.5. On average, the likelihood of the true ges-

ture class was 0.925 (diagonal elements of Figure 2.4(b)), which can be thought of as the “confidence” of the classifier in the correct class. Note that these results are for full, *pre-segmented* gestures, i.e., only full gestures (regardless of the number of strokes) were used for training and testing. While not incorporated in this work, it would be possible (and perhaps desirable) to include partial or incomplete gestures in \bar{G} .

2.2.3 Stroke Classification

The stroke classifier, which returns $p(s_k = c_s | o_{s_k})$, was learned on 720 strokes (80 of each class $c_s \in \bar{S}$) and tested on 1398 strokes. Figure 2.5(a) shows the confusion matrix for the test strokes, which were classified with 88.1% accuracy. Figure 2.5(b) shows the average likelihoods calculated by the learned regression



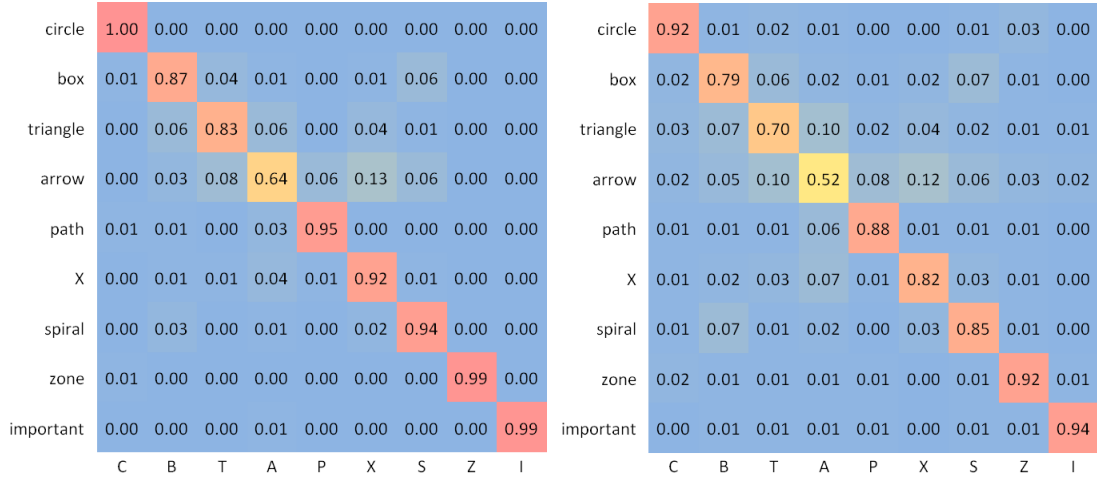
(a) Confusion matrix for gesture classes. Gestures were classified with 98.6% accuracy.

(b) Average observation likelihoods as calculated by Equation 4.5: $p(g_k = c_g | o_{g_k})$

Figure 2.4: Results of classification on 1021 test gestures. Rows correspond to the true class c_g , columns correspond to the recognized class.

model for stroke classes. On average, the likelihood of the correct stroke class was 0.814.

The lower stroke recognition accuracy and corresponding likelihoods indicate that strokes are more difficult to classify than gestures. This is not unexpected, as strokes corresponding to different classes may be very similar. For example, `triangle` and `arrow` are both typically drawn with diagonal straight-line strokes and/or elbow-shaped strokes, and therefore are occasionally misclassified as one another (about 7% of the time). However, an `X` and an `important` are also both generally drawn using straight-line strokes, yet they are rarely misclassified as one another. This is namely because `important` strokes are drawn vertically while `X` strokes are drawn diagonally. Since o_{s_k} encodes a rich set of information, seemingly ambiguous strokes can still be reliably classified.



(a) Confusion matrix for stroke classes. Strokes were classified with 88.1% accuracy.

(b) Average observation likelihoods as calculated by Equation 4.5: $p(s_k = c_s | o_{s_k})$

Figure 2.5: Results of classification on 1398 test strokes. Rows correspond to the true class c_s , columns correspond to the recognized class.

Another cause for the slightly lower recognition accuracy is the diversity among strokes corresponding to a single class. For example, an `arrow` is most commonly drawn with either a single stroke (the full gesture), a straight arrow tail stroke and a bent arrowhead stroke, or three separate straight-line strokes. In the current implementation, these very differently shaped strokes all belong to the same `arrow` stroke class and are used to learn a common set of weight vectors w . Stroke recognition accuracy could possibly be increased if each of these stroke sub-labels were classified separately, however this would either require further hand-labeling or some automatic clustering algorithm during the training phase, as well as a larger training data set.

2.2.4 Interstroke Classification

For the data collected, \bar{I} consisted of six classes : `box-ST`, `triangle-ST`, `arrow-ST`, `X-ST`, `important-ST` and new gesture transition (where “ST” stands for self-transition). Four gestures (`circle`, `zone`, `path`, and `spiral`) were never drawn with more than one stroke in the training set, so the likelihoods of these self-transitions were assumed to be zero. Interstrokes corresponding to new gesture transitions did not specify which gestures were being transitioned to/from; these pairwise transitions could also be learned if desired.

The interstroke classifier, which returns $p(i_k = c_i | o_{i_k})$, was learned on 198 interstrokes (33 of each class $c_i \in \bar{I}$) and tested on 1304 interstrokes. Figure 2.6(a) shows the confusion matrix for the test interstrokes, which were classified with 81.1% accuracy. This lower recognition rate is not surprising; performing clas-

sification on interstrokes is analogous to determining where spaces belong in a line of text by looking only at two adjacent letters at a time. Figure 2.6(b) shows the average likelihoods calculated by the learned regression model for interstroke classes. On average, the likelihood of the correct interstroke class was 0.675, further supporting the fact that interstrokes are relatively ambiguous. The easiest interstroke to recognize is the `important-ST`, due to the consistency with which users tended to sketch the `important` gesture (one long vertical stroke, followed by one small stroke directly underneath). The regularity of the relationship between successive `important` strokes makes the `important-ST` easily distinguishable from the other interstroke classes, and is classified with 98% accuracy.

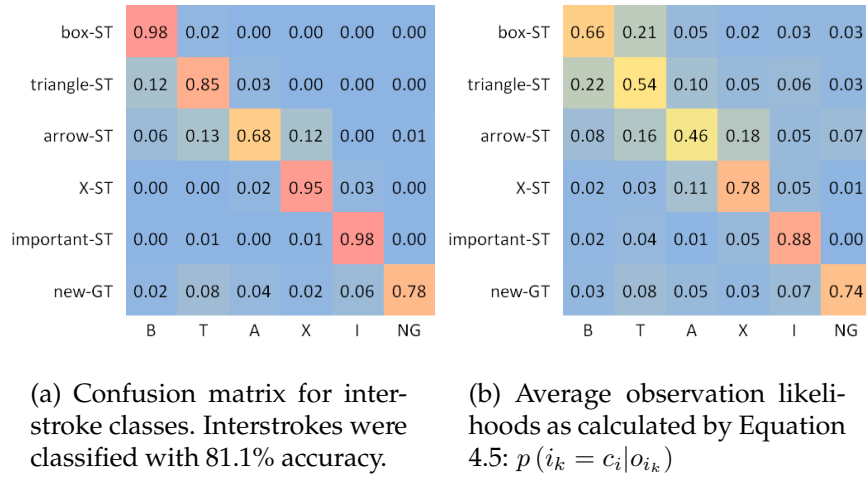


Figure 2.6: Results of classification on 1304 test interstrokes. Rows correspond to the true class c_i , columns correspond to the recognized class.

2.3 Multi-Stroke Sketch Recognition

A key component of the proposed sketch interface is performing on-line recognition of multi-stroke sketches, such as the one shown in Figure 2.7. For robot-control applications, gestures could correspond to desired trajectories, areas to explore or avoid, objects to examine, targets to engage, etc. Therefore, it is important that the sketch be recognized quickly and correctly. In the example shown, some gestures are drawn with multiple strokes, some gestures intersect with other gestures, and one gesture (the path) does not have a pre-defined shape such that template-matching could be used for recognition.

2.3.1 Recognition Algorithm

Using the learned likelihood models from Section 2.2, a forward tree-search algorithm (see Algorithm 2 in Appendix A for details) is implemented to find the most likely sketch by searching for the sequence of strokes, interstrokes and gestures that maximizes Equation 2.1. Figure 2.8 shows an example sketch con-

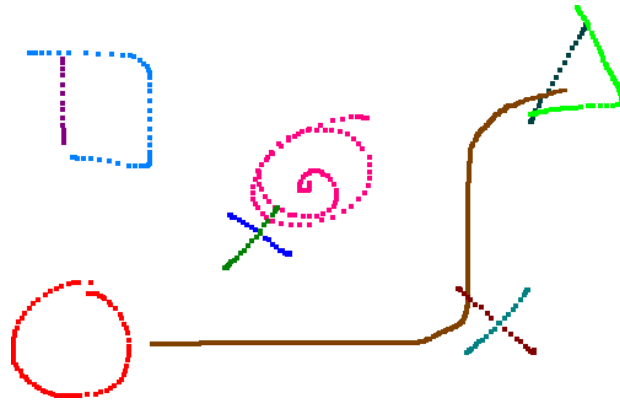


Figure 2.7: An example sketch with seven gestures and eleven strokes (each stroke is drawn in a different color for clarity).

sisting of a single `arrow` gesture drawn with two strokes, and the corresponding search tree. First, $p(s_1 = c_s | o_{s_1})$ is evaluated for the first stroke (arrow tail, drawn in red) for each of the $M = 9$ possible stroke classes c_s . In this case, the node n with the highest likelihood \mathcal{L}_n corresponds to s_1 being an `X`, so this node is expanded first, creating $M + 1$ new child nodes. Next, the first interstroke i_1 and second stroke s_2 (arrowhead, drawn in blue) are considered. If i_1 is an `X-ST`, then s_2 is also an `X` stroke (and s_1 and s_2 compose a complete `X` gesture g_1). If i_1 is a new gesture transition (`NG`) then s_1 creates a complete `X` gesture g_1 , and s_2 belongs to a new gesture g_2 . The likelihood of each child node is calculated incrementally by multiplying the likelihood of its parent node with the latest observation likelihoods using Equations 2.3 and 2.4. This process continues until the leaf node with the highest likelihood represents the last stroke in the sketch. In the example shown in Figure 2.8, the sketch is correctly recognized after exploring 39 nodes. Notice that while the first stroke is most likely an `X`, the full sketch is best explained by assigning both strokes s_1 and s_2 to a single `arrow` gesture.

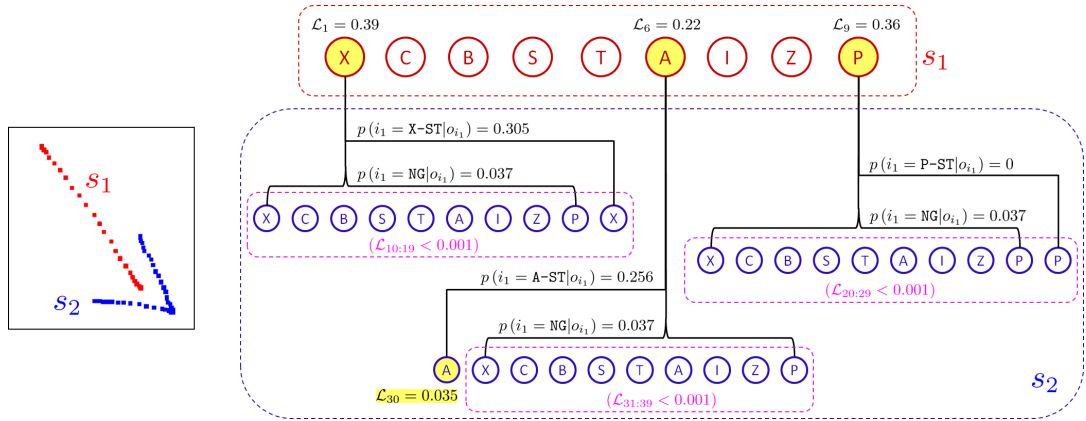


Figure 2.8: (Left) An example sketch (a single `arrow` gesture) composed of two strokes. (Right) The generated search tree.

2.3.2 Modified Likelihood Using Search Heuristic

The space explored in the tree search is, worst-case, $M(M+1)^{N_S-1}$ nodes, where N_S is the total number of strokes in the sketch and M the number of gesture (and stroke) classes, which may be too time consuming to allow real-time evaluation of very large sketches. Unfortunately, as the forward search algorithm progresses through the stroke sequence, the aggregate likelihood of the sketch decreases as numbers ≤ 1 are multiplied (see Equations 2.3 and 2.4). This has the undesirable effect of encouraging the algorithm to perform a breadth-first search, which is worst-case for this constant-depth search tree. Therefore, a heuristic is proposed to ‘penalize’ expanding nodes closer to the top of the tree:

$$\mathcal{L}'_n = \mathcal{L}_n \left(\frac{N_S}{N^{[n]}} \right)^\alpha \quad (2.6)$$

where $N^{[n]}$ is the stroke number of node n and $\alpha \in [0, 1]$. Similar to A* [61], this heuristic results in an *informed* or *greedy* best-first search strategy that expands the node which appears to be most likely to lead towards the goal by estimating the total cost-to-goal. Equation 2.6 encourages a more depth-first-search behavior, but unlike A* the heuristic is not admissible, and therefore sacrifices the guarantee of optimality. Because the search tree is defined such that the depth of the solution is known, other tree-search algorithms such as depth-first iterative-deepening (DFID) and iterative-deepening-A* (IDA*) maintain optimality but do not offer computational savings over best-first-search [62].

For $\alpha = 1$, the proposed heuristic effectively calculates the average node likelihood of the stroke sequence up to the n^{th} node, $l = \mathcal{L}_n \left(\frac{1}{N^{[n]}} \right)$. It then assumes this value for the remaining nodes in the full sequence (up to node N_S), resulting in $\mathcal{L}'_n = l^{N_S} = \mathcal{L}_n \left(\frac{N_S}{N^{[n]}} \right)$. Figure 2.9 illustrates how the stroke number $N^{[n]}$ and

α effect the modified likelihood \mathcal{L}'_n for different values of \mathcal{L}_n . Low-likelihood nodes towards the top of the tree (small \mathcal{L}_n and $\frac{N_S}{N^{[n]}}$) are penalized most aggressively, whereas high-likelihood nodes towards the bottom of the tree (large \mathcal{L}_n and $\frac{N_S}{N^{[n]}}$) are penalized least.

Two metrics are used to evaluate the efficiency and effectiveness of the sketch recognition algorithm: (1) the number of nodes explored in the tree search, and (2) the total number of misclassifications. A misclassification occurs either when a stroke is classified as the incorrect class, or if an interstroke is incorrectly identified as a new gesture transition when it is actually a self-transition (or vice-versa). The number of nodes explored in the tree search indicates computational complexity, and is a key concern for a system that must run in real-time. Specifically, full sketch recognition should be performed at about the same rate as the human can draw (under one second is desirable).

The sketch recognition algorithm was tested on 87 sketches of 5–8 strokes. Figure 2.10(a) plots the average recognition accuracy of the most likely sketch for different values of α . Accuracy is defined as the percent of correctly classi-

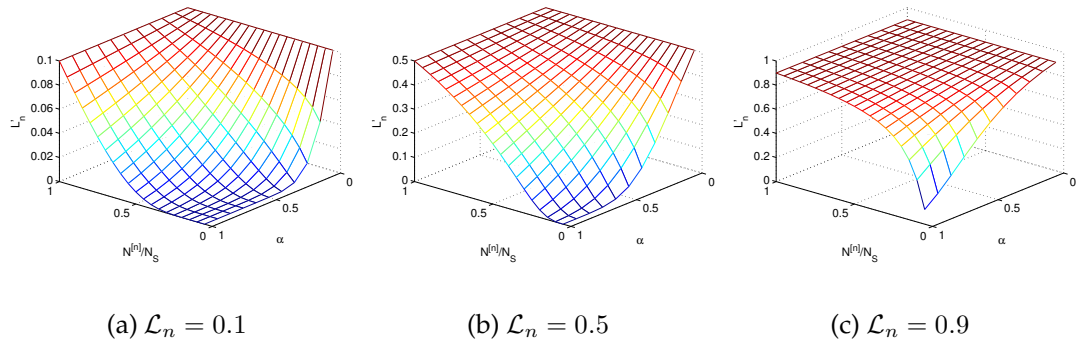
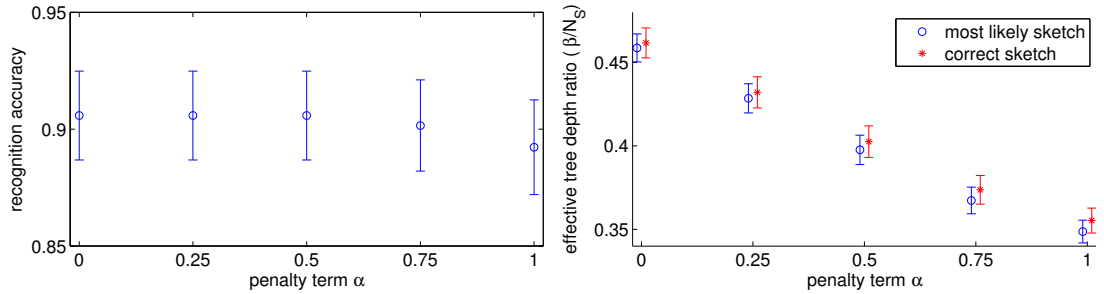


Figure 2.9: Effect of stroke number $N^{[n]}$ and α on the modified likelihood \mathcal{L}'_n . The value of α determines the degree to which nodes are ‘penalized’ in the search.

fied strokes and interstrokes in the most likely sketch. (For example, if a sketch was drawn with 10 strokes and one stroke was classified incorrectly, the sketch accuracy would be 90%.) Figure 2.10(b) plots the *effective tree depth ratio* β/N_S for different values of α where: $\# \text{ nodes searched} = M(M+1)^{\beta-1}$. Here, β represents the size (in number of strokes) of a hypothetical sketch for which the *full* search tree contains the same number of nodes as were explored during sketch recognition. For example, $\beta = 0.5N_S$ means the search algorithm explored the same number of nodes that would exist in the full search tree for a sketch with half as many strokes. The effective tree depth ratio β/N_S can be thought of a measure of the algorithm's efficiency.

Figures 2.10(a)–(b) plot the average recognition accuracy and effective tree depth ratio of the 87 test sketches for different values of $\alpha \in [0, 1]$ in Equation 2.6. These plots show that while increasing α reduces the sketch recognition accuracy slightly (from 90.6% to 89.2%), the effective tree depth ratio decreases by 24%. This represents a computational savings of 72% for a sketch with 5



(a) Average recognition accuracy of the most likely sketch for different values of α . Whiskers are standard errors.

(b) Effective tree depth ratio β/N_S for different values of α (data points are offset for clarity only). The “most likely sketch” is the first found by the search algorithm. Whiskers are standard errors.

Figure 2.10: Algorithm accuracy and efficiency for 87 test sketches of 5–8 strokes for $M = 9$.

strokes and 87% for a sketch with 8 strokes (for $M = 9$). These results suggest that it may be reasonable to incorporate a small penalty term ($\alpha < 0.5$) to help avoid breadth-first-search behavior while sacrificing little accuracy.

With no penalty ($\alpha = 0$), the sketch recognition algorithm yields an average accuracy of 90.6% (the percent of strokes and interstrokes correctly recognized in the sketch). However, of the 87 sketches tested, only 55% were recognized with 100% accuracy on the first try (i.e., the sketch with the highest likelihood) and 75% were correctly recognized by the third try. Clearly, if the sketches are to be used for sending robot commands, it is essential that the *entire* sketch is correctly recognized. To address these errors, users are asked to confirm or reject the recognized sketch. While this may increase operator workload, hypothesis generation and confirmation have been used successfully to avoid problems caused by imperfect recognition [63].

2.4 Experimental Design

A set of experiments was conducted to study the usability of the proposed sketch interface. Sixteen operators each performed four search-and-identify missions by controlling a single remote robot (Fig. 2.11) using two different control architectures: a point-and-click interface using buttons and menus (referred to as “Waypoint mode”), and a sketch interface (“Sketch mode”). While the modes of user input were different, both architectures included the same functionalities. Figure 2.13 shows a screen shot of the interface used. On the left monitor, an overhead view of the environment is displayed, including the robot(s) to be controlled and obstacles detected by the onboard LIDAR. The

right monitor displays the robot camera view(s) and a teleoperation panel (used during Waypoint mode only). Two missions were performed using each interface. Operators were given three primary mission objectives and one secondary objective:

- Primary 1: Explore as much of the environment as possible.
- Primary 2: Locate an unknown number of Points of Interest (POIs) labeled with either a square, triangle, or spiral and add their locations to the map.
- Primary 3: Locate an unknown number of unlabeled POIs and take their picture.
- Secondary: Solve simple addition and subtraction inequalities (see Figure 2.12). Correct answers gained 1 second of fuel, incorrect answers lost 3 seconds of fuel.

Points of Interest (POIs) were large gray trash cans placed at unknown locations throughout the environment (see Fig. 2.13), and could be detected by the

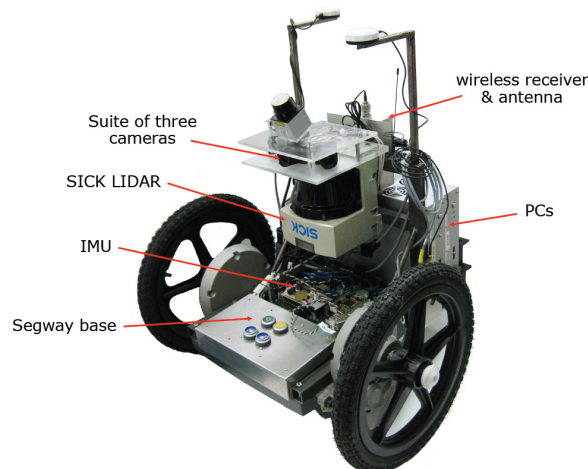


Figure 2.11: Robot used during experiments equipped with an IMU, 180° SICK laser range finder, and three cameras with a combined 150° field of view.

robot with its 180° LIDAR sensor and three cameras with a combined 150° field of view. The three primary objectives represent typical goals of a Search and Rescue operation: explore and map the environment, search for and indicate the locations of victims (labeled POIs), and capture information about potential dangers (unlabeled POIs). The secondary mission objective served two functions. First, it allowed operators to increase the amount of time available for performing the primary mission objectives by earning extra fuel (each mission began with 5 minutes of fuel). Users could solve the inequalities whenever they wanted and were able to, so this also served as a distractor by requiring operators to multi-task and to prioritize simultaneous objectives.

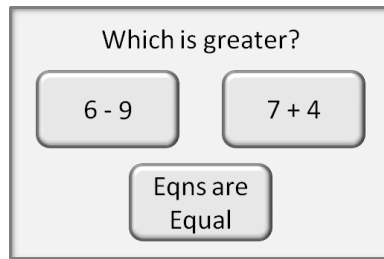


Figure 2.12: GUI panel for answering inequality problems.

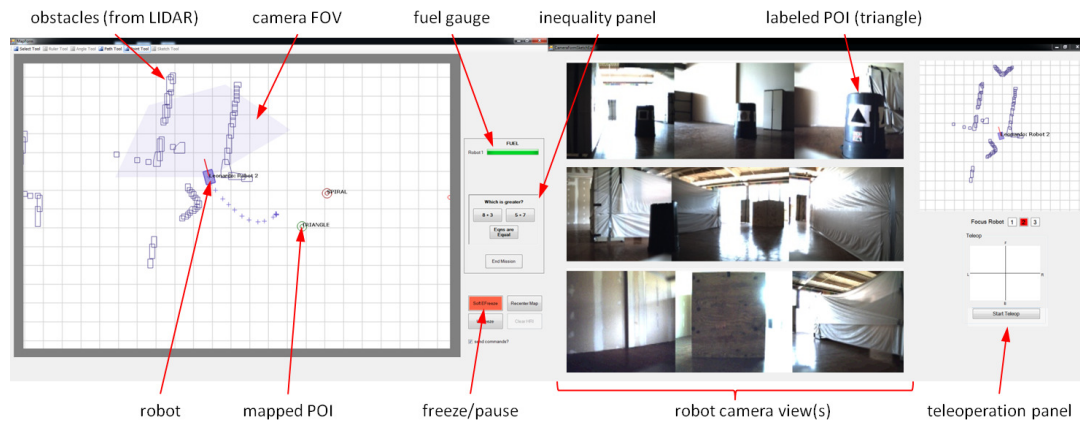


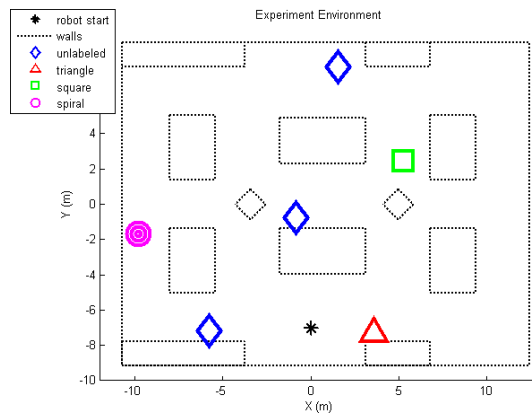
Figure 2.13: Dual-monitor user interface for human-operator experiments. The left monitor displays an overhead view of the robot, obstacles in the environment, and mapped POIs. The right monitor streams the 150° camera field of view.

The $23\text{m} \times 18\text{m}$ experimental environment (shown in Figure 2.14) could not be directly seen by the users from the command station. For each mission there were three unlabeled POIs and three labeled POIs (one of each type); two different POI configurations were used to prevent operators from easily learning the map. The pose of the robot was estimated in real time by integrating odometry and readings from an inertial measurement unit (IMU). Objects in the environment detected by the LIDAR were displayed as wire polygons in the map. The robot was capable of basic obstacle avoidance; if an obstacle could be seen with the LIDAR, it was avoided using a low-level path planning component. Additionally, if the robot was traveling to a waypoint location, the trajectory was planned to avoid all detected obstacles (so long as the waypoint was not placed inside an obstacle, in which case the robot waited for the user to send a new command). The robot could be stopped/paused by pressing the 'Freeze' button on the bottom of the screen at any time, but fuel continued to deplete.

2.4.1 Sketch Command Interface

For the two missions performed using Sketch mode, operators controlled the robot using the nine gestures from Figure 2.3. Users were instructed to sketch "naturally" as if they were trying to communicate with another human, but were otherwise given no instructions as to how the gestures should be drawn. Sketches could be edited by deleting the last stroke or clearing the entire sketch.

The commands corresponding to each gesture are as follows. A circle drawn around a robot selects the robot for future control. An \times sets a single waypoint to which the robot must go, the specific path to be determined by a



(a) Overhead schematic of one environment configuration, showing walls, points of interest (POI), and robot starting location.

(b) View of experimental environment, showing large rectangular obstacles and a square POI.

Figure 2.14: The $23\text{m} \times 18\text{m}$ experimental environment, which was not visible to the operators from the command station. Two POI configurations were used to prevent operators from easily learning the map.

low-level path planner to avoid collisions. An `arrow` commands the robot to travel to the base of the arrow and turn to point in the direction indicated by the arrow head. `Paths` define a desired robot trajectory by setting a series of waypoints 1m apart along the sketched path. A `zone` indicates a specific area that should be explored, and the robot visits randomly sampled points inside the specified zone. An `important` instructs the robot to take a picture of its current camera view. Lastly, `boxes`, `triangles` and `spirals` represent the three types of POIs the robot can encounter. When one of these three gestures is drawn, a marker is added to the map at the gesture centroid location.

When a user finishes drawing a complete sketch, he or she presses the ‘Send Sketch’ button, triggering the recognition algorithm to search for the most likely sketch. Once finished (typically < 1 sec), the recognized sketch is displayed on

the screen for the user to inspect. If the sketch is correctly recognized, the user presses the 'Correct' button. Otherwise, pressing the 'Incorrect' button signals to display the sketch with the next-highest likelihood. Upon pressing the 'Correct' button, the robot executes the appropriate commands based on the recognized gestures, overriding any previous commands that may not have yet been completed.

Figure 2.15 illustrates the sequence of events for an example sketch. In the top left panel, the user inputs a sketch consisting of `circle`, `path`, and `triangle` gestures. In the top right panel, the interface displays the most likely sketch and waits for the operator to confirm. Finally, the bottom two panels show the mapped POI and the newly-selected robot following the instructed path after user confirmation.

2.4.2 Waypoint Command Interface

For the two missions using Waypoint mode, three tools were available to the user during these missions: Select Tool, Path Tool, and Point Tool. Tools could be activated or deactivated either by clicking on the corresponding button at the top of the screen, or by right-clicking and navigating through a pop-up menu. Select Tool is used to select or deselect a robot by clicking and dragging a box around it. Path Tool is used to define a desired trajectory using one or more waypoints. To command a selected robot to follow a constructed path, the operator right-clicks on the map and chooses "Send Path" from the top of the pop-up menu. Point Tool is used to map the locations of labeled POIs by right-clicking and choosing "Add new spiral" (for example) in the pop-up menu and left-

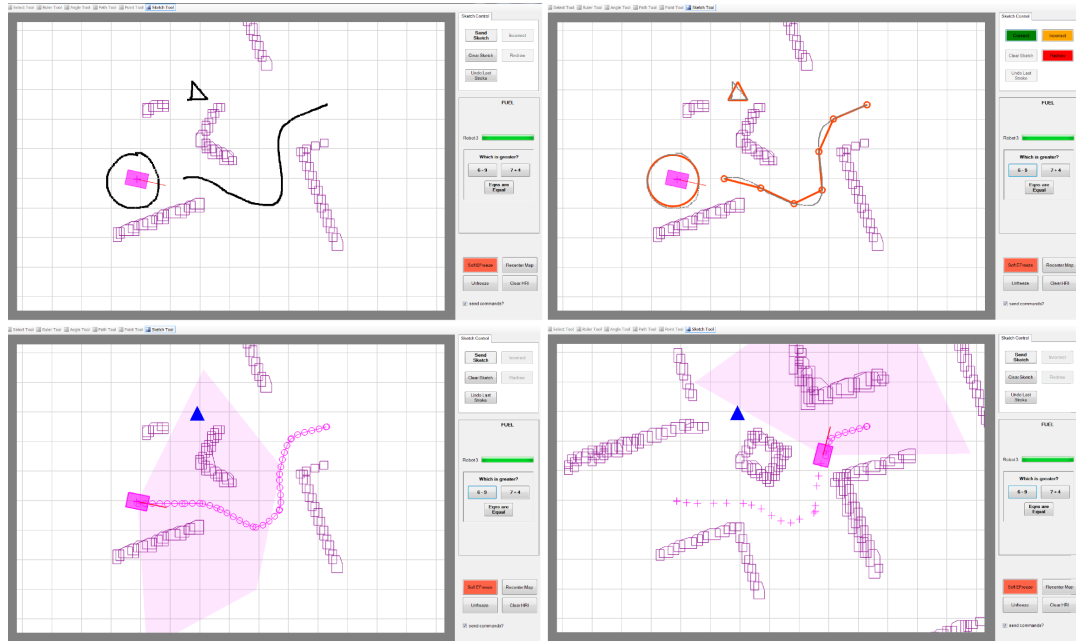


Figure 2.15: Sequence of events for an example sketch. The robot is displayed as a rectangle with a line pointing out the front (a selected robot also displays its 150° 3-camera field of view). Obstacles observed with the 180° laser range finder are displayed as wire polygons on the map. In the top left panel, the user sketches three gestures: circle, path, and triangle. After the user presses the ‘Send Sketch’ button, the recognized sketch with the highest likelihood is rendered in orange, superimposed on the original sketch (shown in top right panel). Upon confirmation from the user, the robot in the circle is selected and follows the commanded path, and the triangle POI’s location is placed on the map (bottom panels).

clicking to place the marker on the map. Point Tool is also used to take a picture of unlabeled POIs by right-clicking and choosing “Take a picture” in the pop-up menu and left-clicking to take the picture.

The robot can also be teleoperated directly using a panel enabling joystick-like control. This is especially useful for maneuvers such as turning the robot in place, which can not be achieved using Path Tool. Teleoperation overrides all other controls, including the robot’s automatic obstacle avoidance, so operators

were advised to use teleoperation with caution.

2.5 Experimental Results

2.5.1 Mission Objectives

Four performance metrics were used to evaluate how well operators performed the mission objectives: (1) percent of environment explored, (2) number of labeled POIs places correctly on the map, (3) number of unlabeled POIs successfully photographed, and (4) number of inequalities solved. Each metric was evaluated at the five-minute mark (the original time allotted at mission start) and at the final mission time (accounting for any fuel gained/lost as a result of solving inequality problems). Figure 2.16(a) plots the average environment exploration for each interface. This was calculated by dividing the environment into 100 cells and counting how many had been visited by the robot. Some cells were inside obstacles, therefore 100% exploration was not possible. Results show that there was not a significant difference in the area explored using either interface.

Figure 2.16(b) plots the percent of total POIs found and correctly mapped (in the case of labeled POIs) or successfully photographed (for unlabeled POIs). A labeled POI was considered correctly mapped if the appropriate marker was placed within 1m of the object's true location. Operators were able to find and map approximately the same number of labeled POIs using either interface, but successfully photographed 8.3% more unlabeled POIs in Sketch mode than Waypoint mode (however, this difference is not statistically significant). Upon

closer examination of the data, it was observed that operators often attempted to take a picture while the robot was still moving. This would happen when the robot was following a path or waypoint command while the operator noticed the POI in the camera view. Many times, the operator was not quick enough and the POI left the camera view before the user was able to take the picture using Waypoint mode. This indicates that Sketch mode may lend itself better to rapid task-switching. Even though the missions required issuing several different types of commands, Sketch mode allowed users to operate within a single framework; users did not have to remember what mode they were currently in, or “switch gears” to perform a different task. For some users, this may have enabled the time-sensitive task of photographing unlabeled POIs to be performed more successfully in Sketch mode than Waypoint mode.

The average number of inequalities solved is shown in Figure 2.17(a). Operators using Sketch mode were able to solve an average of 53 inequalities per

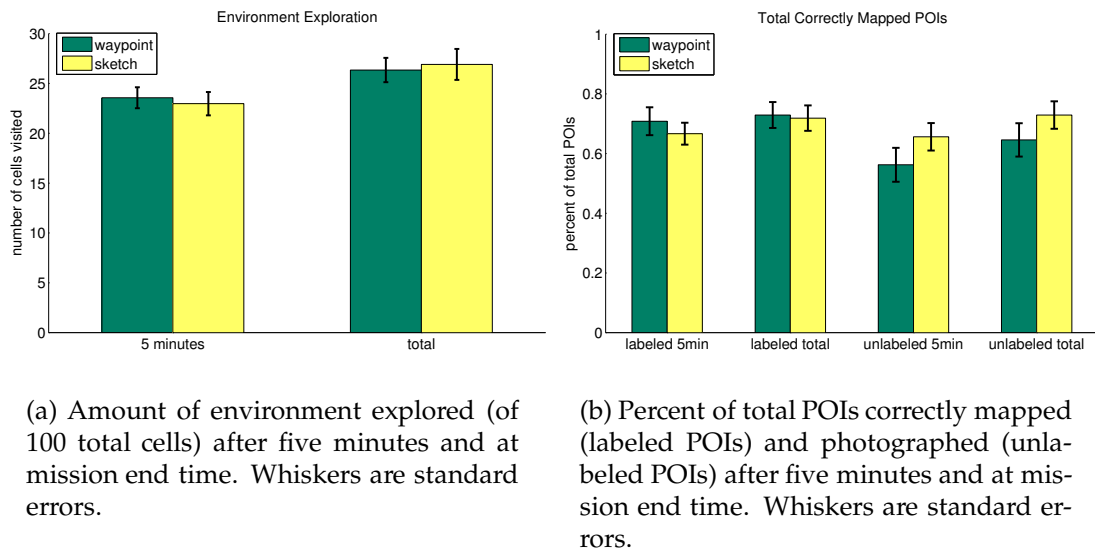
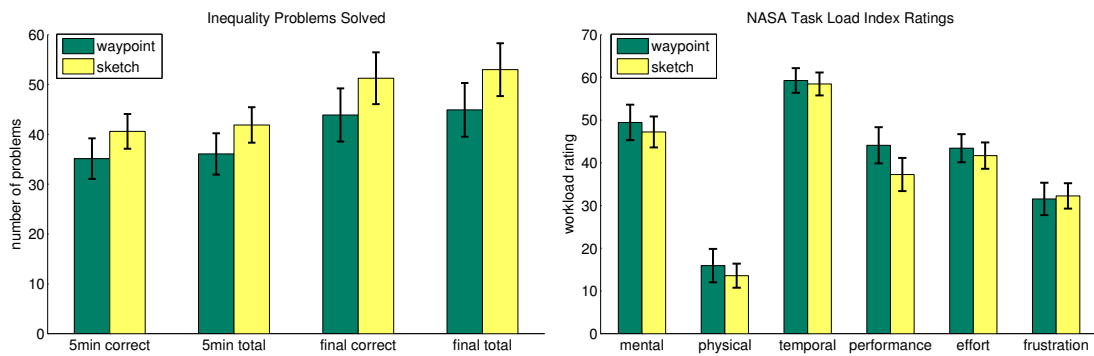


Figure 2.16: Operator performance for primary mission objectives.

mission (51 correctly), and earned an extra 45 seconds of fuel. Operators using Waypoint mode solved 45 inequalities per mission (44 correctly), and earned an extra 41 seconds of fuel. While this difference was not statistically significant, it indicates that operators may be better able to multi-task and more easily sustain a higher workload without sacrificing the primary mission objectives using Sketch mode.

2.5.2 Subjective Operator Feedback

Subjective measures of workload were obtained utilizing the following six sub-scales of the NASA-Task Load Index (NASA-TLX) [64]: mental demand, physical demand, temporal demand, performance, effort, and frustration level, as described in Table 2.1. Participants were asked to rate each of the sub-scales after each mission, and the results are shown in Figure 2.17(b). In all six categories, operators reported that they experienced approximately equal workload



(a) Average number of inequalities solved (correct and total) after five minutes and at mission end time.

(b) Average NASA-TLX survey ratings (1–100). Whiskers are standard errors.

Figure 2.17: Operator performance for secondary mission objective and subjective workload measures.

Table 2.1: Explanation of NASA-Task Load Index sub-scales. Each scale was rated from 1–100.

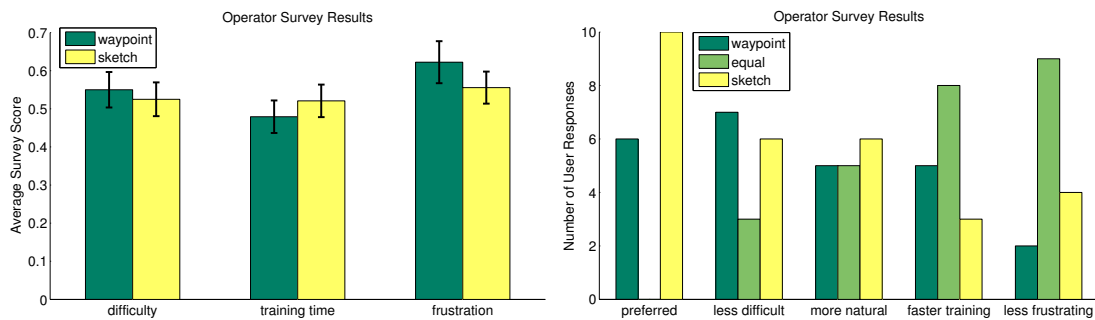
TLX Sub-Scale	Description
Mental Demand	The amount of mental and perceptual activity required (e.g., thinking, deciding, calculating, remembering, looking, searching, etc.)
Physical Demand	The amount of physical activity required (e.g., pushing, pulling, turning, controlling, etc.)
Temporal Demand	The amount of pressure the user felt due to the rate or pace at which the tasks occurred
Effort	The amount of effort (mentally and physically) required to accomplish the user’s level of performance
Performance	How successful the user was in accomplishing the goals of the task and how satisfied the user was with his/her performance
Frustration	How insecure, discouraged, irritated, stressed and annoyed (versus secure, gratified, content, relaxed and complacent) the user felt during the task

between Sketch mode and Waypoint mode.

After completing all missions, each operator was also asked to fill out an optional survey to give his/her feedback on the experiment and the interfaces. Figure 2.18 plots the responses to questions concerning how difficult/natural/frustrating each interface was, how much training time was required before the operator was comfortable with the interface, and which interface was preferred. These plots indicate that although Sketch mode required more training time, it was also less frustrating to use than Waypoint mode (although not statistically significantly so). A longer training time for Sketch mode is somewhat expected, since Waypoint mode was similar to more familiar in-

interfaces such as computer and video games. Figure 2.18(b) is a histogram of operator responses for pairwise comparisons between the two interfaces. Overall, 10 of the 16 operators reported that they preferred using Sketch Mode over Waypoint mode.

In response to the question, “How accurate/fast was the program at recognizing the sketches you drew?” 81% (13 of 16) of users felt their sketches were recognized well or very well, and 94% (15 of 16) felt the speed of recognition was good or very good. In general, users whose gestures were reliably recognized on the first try greatly preferred using Sketch mode. A few operators whose gestures were not easily recognized, requiring the ‘Incorrect’ button be pressed several times or the user to re-draw the gestures, preferred Waypoint mode. Since the HMM distributions were learned on a different set of people, poor recognition could likely be improved by incorporating an algorithm training phase for each user, or by re-learning the model parameters on-line as the



(a) Average survey scores (0–1) reported by operators, where lower scores are “better”. Whiskers are standard errors.

(b) Number of operator survey responses comparing the two interfaces. Responses of “equal” indicate that the operator did not observe a noticeable difference between interfaces.

Figure 2.18: Post-experiment survey responses. 10 out of 16 users preferred Sketch mode over Waypoint mode.

operator interacts with the interface.

It is possible that users would have preferred using a pen-tablet rather than a mouse in Sketch mode, or that training time and/or sketch recognition accuracy could have been affected (either positively or negatively) with a tablet interface. However, since the specific input device is not the main focus of this work, a standard mouse was employed in both modes. This was done to ensure that the users' performance and/or experiences were affected by the interface strategy alone, and not the use of a new and possibly less familiar input device. The potential effects of a pen-tablet on mission performance and user preference perhaps warrants further study.

Many of the operators expressed the opinion that the main benefit of Waypoint mode was the fact that it was deterministic, but that it was difficult to navigate the menus to switch quickly between multiple modes. A few users, specifically those with a lot of video game experience, suggested that Waypoint mode could be improved by using 'hot keys' or combinations of keys (using Ctrl, Shift, etc.) to switch between modes, rather than buttons and menus (even though such an interface would likely require significant more training). Several operators expressed the opinion that both interfaces could be improved if they were somehow merged, utilizing the strengths of both. For selecting/deselecting the robot, most users desired the speed and precision offered by Waypoint mode; for commanding paths and placing POI locations, Sketch mode was the preferred interface. Another improvement that operators suggested for Sketch mode was to display the top two or three likely sketches at once, and allow the user to choose the correct one, rather than cycling through by clicking a 'Correct' or 'Incorrect' button.

2.6 Chapter Conclusions

In this chapter, a fully probabilistic sketch recognition interface for natural robot control was proposed. Sketches are modeled using a variable duration hidden Markov model with distributions learned from training data, allowing the interface to be customized to the intended users' drawing styles and shift the burden of recognition from the operator to the machine. Because gestures are not pre-defined (such as straight lines or closed polygons), the framework can support arbitrary gestures. A set of 94 gesture features, 91 stroke features, and 9 inter-stroke features are defined that yield average recognition accuracies of 98.1%, 88.1% and 81.1%, respectively. The classifiers are used in a forward search algorithm to find the combination of gestures, strokes, and interstrokes that maximizes the full sketch likelihood. A heuristic is implemented to discourage breadth-first search behavior, which decreases the average computational cost by over 72% on a set of 87 test sketches but sacrifices the guarantee of optimality. The most likely sketch is displayed to the operator, who then confirms or rejects the recognized sketch.

A set of indoor search-and-identify missions was conducted in which operators controlled a single mobile robot using the proposed sketch interface and a point-and-click menu interface. On average, operators were equally effective at satisfying the mission objectives using the two interfaces. User surveys conducted at the end of the experiments indicate that most operators preferred using sketch control and felt that sketch recognition was performed quickly and accurately, suggesting that this is an effective approach for mobile robot control. Users reported that the biggest advantage of the sketch interface was the ability to easily perform heterogeneous tasks within a single command framework.

There are several benefits and possible extensions to the proposed approach for mobile robot control. First, since discriminating gesture attributes are learned and not hard-coded, new commands can be easily incorporated (or obsolete commands removed) by re-learning the model using a new set of training data. During the training phase, the supervised learning algorithm automatically chooses the combination of features that best distinguishes gestures from one another, maintaining high accuracy even as gestures are added or changed. While not implemented in this work, this also enables the model to be updated on-line as the operator interacts with the interface via the hypothesis generation and confirmation routine. Another benefit is that sketch recognition is probabilistic, so there is a notion of “how certain” the algorithm is with each classification. This is not only necessary for performing an efficient informed search, but this information also could be used to determine when user confirmation is actually required. Rather than requiring confirmation of every sketch (which can be monotonous and distracting to the user), the system could instead prompt the user for help resolving ambiguities between comparatively-likely sketches. Lastly, the sketch interface allows operators to communicate in a way that other human agents can readily understand; “playbook” style sketching is often an easy and reliable way for humans to communicate strategies and intentions with one another. The sketch interface not only allows the human issuing commands to operate in a familiar and natural manner, but it also promotes a single mode of interaction with both human and robotic agents.

CHAPTER 3

A QUALITATIVE PATH PLANNER FOR ROBOT NAVIGATION USING HUMAN-PROVIDED MAPS

This chapter presents an algorithm for controlling a mobile robot using a qualitative map consisting of landmarks and path waypoints, such as one sketched by a human. The sketch need not contain precise information about the entire environment, but only information relevant to the navigation task. Additionally, the sketched map may not be quantitatively accurate, but should be *qualitatively* similar to the true environment. “Qualitative” here refers to the preservation of spatial relations of landmarks and waypoints [65], such that the map can be understood and used by another human. A quadratic optimization scheme is used to calculate the location of waypoints in the environment that best maintain their spatial relationships to observed landmarks as specified by the sketched path. This approach provides an online, dynamic planner that accounts for localization errors (i.e., uncertainty in the true landmarks’ positions) as well as missing or unobserved landmarks; can accommodate complicated trajectories (e.g., not limited to straight-line paths); and is robust to sketch distortions. This chapter expands upon [66] by introducing a method for waypoint extraction along the sketched path, analyzing the effects of sensor constraints and landmark sparseness on planner performance, and implementing the planner in a set of human trial experiments.

In contrast to previous approaches discussed in Section 1.2, the method described in this work does not require extraction and matching of local landmark templates. The proposed algorithm makes use of probabilistic mapping strategies such as Simultaneous Localization and Mapping [67] to plan arbitrarily-

shaped trajectories with respect to uncertain landmarks. By maintaining a map of estimated landmark locations, the robot plans an informed trajectory through the environment, rather than implementing a reactive planner triggered by discrete local observations. Furthermore, although the explicit definitions of qualitative spatial relationships in the environment (such as ‘near’ or ‘between’) are not required, the algorithm aims to preserve such relationships.

The use of local landmarks for controlling mobile robot navigation is motivated by how humans naturally communicate and execute course instructions. In [68], five experiments were conducted to examine different approaches to remembering, communicating, and following verbal route directions, such as presenting the directions in correct temporal-spatial order, and using different ‘choice points’ (i.e., landmarks). Results from three of the experiments validated the practices of including descriptives and concentrating delimiters at choice points, as a way of maintaining ‘common ground’ and reducing uncertainty along the route. In [28], two studies intended to identify the cognitive functions of landmarks were performed. Results showed that landmarks are generally considered to be key components for constructing the representations used for navigation, primarily for specifying re-orientation locations, helping to locate other landmarks, and confirming the navigator’s situation along the route. For example “to the north,” “between” and “to the right” are natural language descriptions that may be used to place a waypoint with respect to specific landmarks. By taking advantage of how humans naturally communicate navigation instructions, the proposed method provides a flexible and intuitive approach to mobile robot control.

This chapter is organized as follows. Section 3.1 introduces the concept of

navigation using a qualitative map and presents metrics for evaluating planner performance. Section 3.2 presents the Qualitative Path Planner (QPP) algorithm, which mimics how humans naturally navigate by optimizing for locations that ‘best’ maintain appropriate spatial relationships to observable landmarks in the environment. The algorithm also includes a method for extracting waypoints at appropriate locations along the sketched path, and a means for estimating the transformation between the sketched map and the estimated map of the true environment. Practical implementation issues are also discussed, such as sparse landmarks and limited sensor field of view. Several experiments are presented in Section 3.3, including representative examples that distinguish the qualities of the QPP from previous approaches. A sensitivity study is performed, illustrating the robustness of the algorithm to map inaccuracies and susceptibility to robot sensor constraints. Lastly, the QPP is implemented in human trial experiments, during which novice users commanded a mobile robot to navigate to a goal using sketched maps. Mission success is defined as the distance between the final robot location and the desired goal, and is evaluated as a function of the number of landmarks in the sketch, the relative sketch distortion, the defined path-to-goal, and the robot sensor range. Section 3.4 presents final discussions and conclusions.

3.1 Navigation Using a Qualitative Map

3.1.1 Concept and Architecture

Figure 3.1 illustrates an example navigation task for which an approximate map (top) is used to provide route instructions through an environment for which

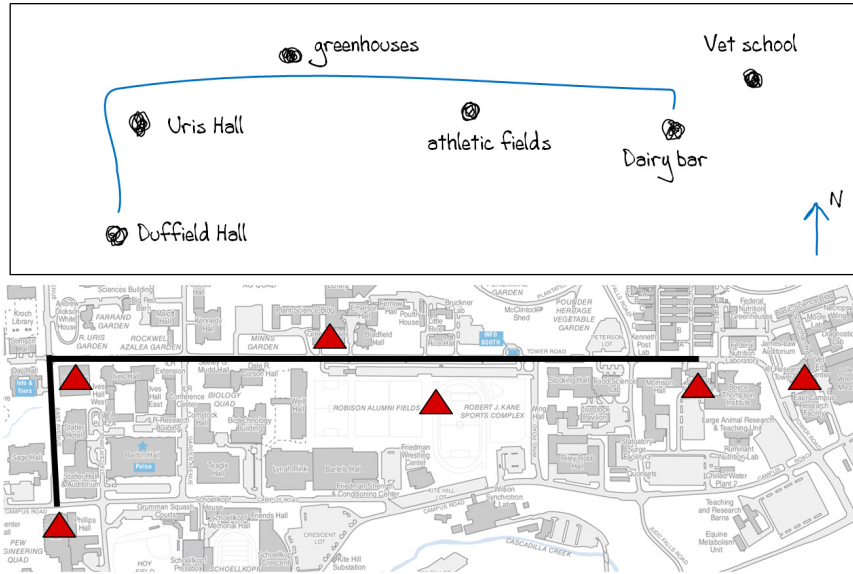


Figure 3.1: (top) Human-provided map of the environment with desired path drawn in blue. (bottom) True map of the environment, which the robot does not have access to. True landmark locations are shown as red triangles.

a true map (bottom) is not available. In the qualitative map shown in Figure 3.1(top), a human has sketched on a tablet PC a path from Duffield Hall to the Cornell Dairy Bar with respect to six landmarks on the Cornell University campus. To another person, the navigation task is quite simple and well-specified: the agent starts traveling north from Duffield Hall, turns right after passing Uris Hall on the right, passes the greenhouses on the left, passes the athletic fields on the right, and arrives at the Dairy Bar on the right; if the agent reaches the Vet School, it has gone too far. Most reasonable people would be able to reach the Dairy Bar using only this qualitative map, even though it does not exactly match the truth, shown in Figure 3.1(bottom). The goal of the proposed Qualitative Path Planner (QPP) is to allow a robot agent to similarly navigate its environment using only a sketched map and measurements to observable landmarks.

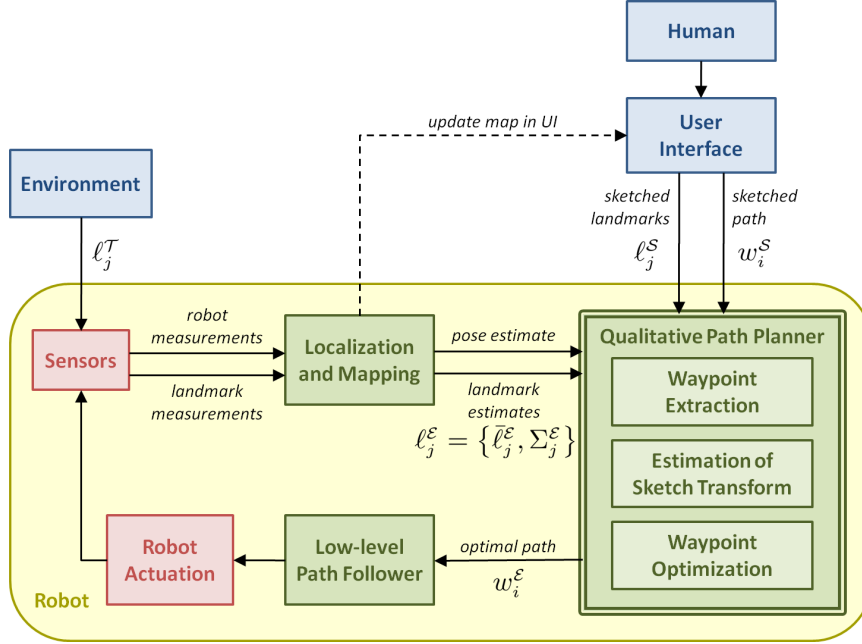


Figure 3.2: A block diagram of the system architecture. A robot receives commands from a human user in the form of an approximate map of landmarks and a desired path. The QPP solves for waypoints in the estimated environment that best maintains appropriate spatial relationships to the observed landmarks.

Figure 3.2 shows a diagram of the system architecture considered in this work. A mobile robot is equipped with (noisy) sensors that measure its pose with respect to some known coordinates, as well as the locations of landmarks in its environment, where a ‘landmark’ is any environmental feature that can function as a point of reference. A human user inputs approximate locations of these landmarks and desired path waypoints via a qualitative map on a PDA, Tablet PC or other computer interface. These are fed to the QPP along with the most recent robot pose estimate and observed landmark locations. The planning algorithm, described in Section 3.2, optimizes for path waypoint locations in the estimated environment that best maintain qualitative spatial relationships to the landmarks corresponding to those on the sketched map. The robot then implements a low-level path follower to reach the desired waypoints.

The data association problem is not considered here, i.e., when the robot observes a landmark in the environment, it is assumed that the robot knows to which sketched landmark (if any) the measurement corresponds. In practice, this may be accomplished using various estimation techniques for locating and identifying unambiguous objects [69, 70, 71]. All examples presented here consider range and bearing measurements made with respect to robot body coordinates, such as those from a laser range finder or stereo vision system.

3.1.2 Performance Metrics

The goal of the QPP is to plan a trajectory that preserves spatial relations of landmarks and waypoints in the corresponding environment. In a natural-language context, these spatial relations may be propositional constraints such as, ‘near the chair,’ ‘between the wall and the table,’ and ‘around the tree.’ The constraints may also include more specific quantities or directions, such as ‘*about 5 meters* from the chair,’ ‘*halfway* between the wall and the table,’ and ‘around the tree *to the north*.’ While these concepts are inherently difficult to quantify, the following performance metrics are used to capture the ‘soft’ nature of the evaluation:

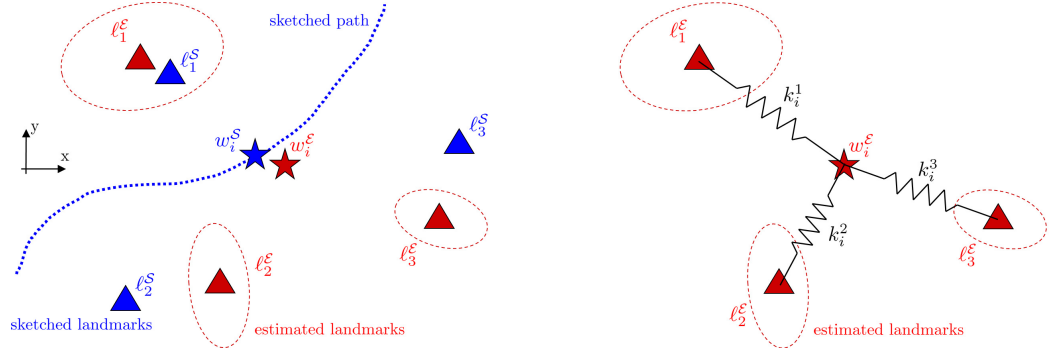
- Gate traversal: Does the trajectory pass between pairs of neighboring landmarks where appropriate?
- Closeness: Does the trajectory come ‘too close’ to (or collide with) a landmark?
- Relative passing: Does the trajectory pass by landmark(s) on the correct side (e.g., landmark is to the right of the robot)?

- Goal offset: How close does the robot get to its desired goal point?
- Solvability: Does a solution exist? How quickly/easily can a solution be found?

These metrics are used to evaluate the performance of the proposed QPP under different experimental conditions such as landmark sparseness and map distortion, as well as to compare the algorithm to other qualitative planning approaches.

3.2 Qualitative Path Planner (QPP)

Figure 3.3 illustrates the concepts and notation of the sketched and estimated maps, as used in the QPP. A 2D sketched map is comprised of path waypoint locations $w_i^S, i = 1 \dots n$ and point landmark locations $\ell_j^S, j = 1 \dots m$, where S denotes data generated from the human sketch. As the robot collects observations of landmarks in the true environment ℓ_j^T (T denotes truth), an estimated map of Gaussian-distributed landmark locations $\ell_j^E = \{\bar{\ell}_j^E, \Sigma_j^E\}$ is generated, where E denotes estimates generated from a filter (see Localization and Mapping block in Figure 3.2). The proposed QPP calculates the location of a desired trajectory waypoint w_i^E in the estimated map/environment, corresponding to sketched waypoint w_i^S . In this work, all landmarks are assumed to be static and the robot is operating in a two-dimensional environment, however, the approach proposed can be extended to dynamic environments and higher dimensions.



(a) A sketch map (blue) superimposed on the estimated map (red), where the landmarks ℓ_j^S do not align perfectly with ℓ_j^E .

(b) Estimated landmarks ℓ_j^E connected by linear springs k_i^j to the proposed waypoint location w_i^E .

Figure 3.3: For each landmark, the spring constant k_i^j is calculated as a function of the distance between waypoint w_i^S and landmark ℓ_j^S , and the uncertainty in the estimated location of the landmark Σ_j^E .

3.2.1 Waypoint Optimization using Potential Energy

Since the sketched map is typically not an exact representation of the observed environment, ℓ_j^S and $\bar{\ell}_j^E$ are not perfectly aligned with each other, as in Fig. 3.3(a). In order to maintain appropriate spatial relationships within the environment, the QPP connects each waypoint w_i^E to each landmark ℓ_j^E by a virtual linear spring with spring constant k_i^j , as shown in Figure 3.3(b), which attempts to ‘hold’ the waypoint at the same relative position as indicated on the sketched map (i.e., the spring for landmark j is unstretched when $w_i^E - \bar{\ell}_j^E = w_i^S - \ell_j^S$). Each spring constant k_i^j affects the degree to which landmark j influences the location of estimated waypoint w_i^E , and is calculated as a function of the Euclidean distance between the sketched waypoint w_i^S and sketched landmark ℓ_j^S , and the uncertainty in the location of landmark j , Σ_j^E . Here, k_i^j at time t is defined as:

$$k_i^j(t) = \left[(d_i^j)^p |\Sigma_j^E(t)|^{1/2} \right]^{-1} \quad (3.1)$$

where $d_i^j = \|w_i^S - \ell_j^S\|$ and $p \geq 0$. (Note that the units of the sketched map are assumed to be scaled to match those of the environment; this can be defined *a priori* or can be estimated online using the method described in Section 3.2.3.) While other factors could be included in the definition of k_i^j (for example, a distant mountain or building may be a useful landmark for determining bearing), Equation 3.1 is appropriate for applications in which landmarks are intended as local (short-range) reference points along the path, which is typical of how humans navigate and identify landmarks as route descriptors [28].

In Equation 3.1, k_i^j decreases when the uncertainty in the location of landmark j is high, as specified by $\Sigma_j^\mathcal{E}$. Physically, $|\Sigma_j^\mathcal{E}(t)|^{1/2}$ is proportional to the volume of the corresponding uncertainty ellipsoid. This term discourages the planning of trajectories relative to landmarks that haven't been localized well (or at all). Additionally, k_i^j decreases as the distance between w_i^S and ℓ_j^S increases (i.e., the location of a waypoint is more heavily influenced by closer landmarks in the sketch). As $p \rightarrow \infty$, only the landmark closest to the waypoint in the sketched map is considered, and all others are ignored. In this work a p value of 2 is used, making $k_i^j \propto (d_i^j)^{-2}$. This choice of p is motivated by physical forces such as gravitational, magnetic and electrostatic forces, which are inversely proportional to the square of the distance between masses, magnetic poles, and point charges, respectively.

The estimated location of the i^{th} waypoint $w_i^\mathcal{E}$ is found by minimizing the potential energy stored in all m springs:

$$U_i = \frac{1}{2} \sum_{j=1}^m k_i^j \|(w_i^S - \ell_j^S) - (w_i^\mathcal{E} - \bar{\ell}_j^\mathcal{E})\|_2^2 \quad (3.2)$$

The point $w_i^\mathcal{E}$ that minimizes Equation 3.2 best maintains its spatial relationships

to all landmarks (proportional to k_i^j). The total potential energy in all the springs connected to the i^{th} waypoint $w_i^\mathcal{E}$ can be minimized using a quadratic program:

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} [\mathbf{w}^T H \mathbf{w} + \mathbf{c}^T \mathbf{w}] \quad (3.3)$$

where, in two dimensions:

$$\mathbf{w} = w_i^\mathcal{E} = \begin{bmatrix} w_{i,x}^\mathcal{E} \\ w_{i,y}^\mathcal{E} \end{bmatrix}$$

$$H = \begin{bmatrix} \sum_{j=1}^m k_i^j & 0 \\ 0 & \sum_{j=1}^m k_i^j \end{bmatrix}$$

$$\mathbf{c} = \begin{bmatrix} 2 \sum_{j=1}^m k_i^j (\ell_{j,x}^\mathcal{S} - w_{i,x}^\mathcal{S} - \bar{\ell}_{j,x}^\mathcal{E}) \\ 2 \sum_{j=1}^m k_i^j (\ell_{j,y}^\mathcal{S} - w_{i,y}^\mathcal{S} - \bar{\ell}_{j,y}^\mathcal{E}) \end{bmatrix}$$

If there are no additional constraints, a unique global solution to this optimization problem is guaranteed to exist and can be solved in closed form:

$$w_{i,x}^\mathcal{E} = \frac{\sum_{j=1}^m k_i^j (w_{i,x}^\mathcal{S} + \bar{\ell}_{j,x}^\mathcal{E} - \ell_{j,x}^\mathcal{S})}{\sum_{j=1}^m k_i^j} \quad (3.4)$$

$$w_{i,y}^\mathcal{E} = \frac{\sum_{j=1}^m k_i^j (w_{i,y}^\mathcal{S} + \bar{\ell}_{j,y}^\mathcal{E} - \ell_{j,y}^\mathcal{S})}{\sum_{j=1}^m k_i^j} \quad (3.5)$$

In its final implementation, the optimization is performed at each time step to update the current ‘best’ location of $w_i^\mathcal{E}$ in the estimated environment. The quadratic program in Equation 3.3 can be solved for each of the $i = 1 \dots n$ waypoints in the path, or just the next untraversed waypoint.

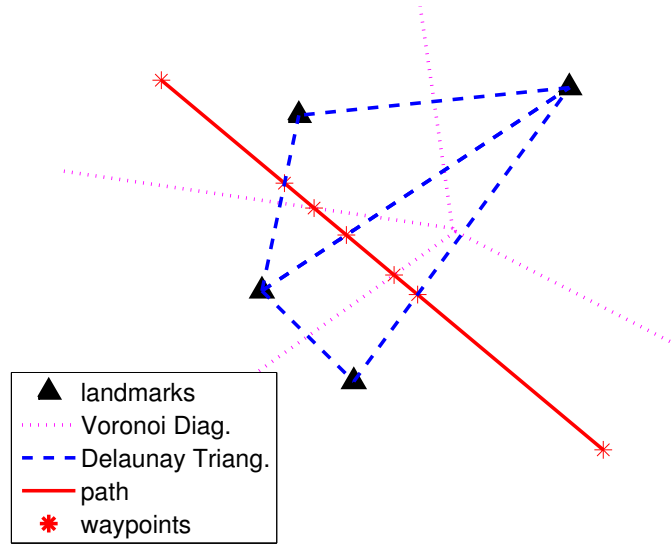


Figure 3.4: Waypoints (*) are extracted at points along the path that intersect with lines in the Voronoi-Delaunay graph.

3.2.2 Waypoint Extraction using Voronoi-Delaunay Graph

Many previous approaches to qualitative navigation define waypoints along the path at locations where the control or orientation of the robot must change. For example, [24] extracts ‘critical nodes’ as points along the sketched route that correspond to a change of orientation. In [34], ‘via points’ are defined by the human, which are assumed to be connected by straight-line obstacle-free paths. [72] encodes route descriptions as a series of routers that express landmarks to be passed and places at which an orientation change is required.

In contrast to these approaches, the QPP defines waypoints as points along the sketched path that intersect with the Voronoi-Delaunay graph [73] created by the sketched landmarks; an example is shown in Figure 3.4. The use of the Voronoi-Delaunay graph to extract waypoints is chosen in part because it mimics how humans navigate [28]. Specifically, this method selects locations along

the path that can be qualitatively described with respect to two ‘nearby’ landmarks (namely, *between* and *equidistant to*).

Algorithm 1: Pseudo-code for waypoint extraction using the Voronoi-Delaunay graph (VD).

```

 $w_{1:n}^S$  defined by human user
 $W^S = \emptyset$ 
for  $i = 1 : n - 1$  do
     $W^S = W^S \cup w_i^S$ 
    for all  $(x, y) \mid (x, y) \in L_i \wedge (x, y) \in \text{VD}$  do
         $W^S = W^S \cup (x, y)$ 
    end for
end for
 $W^S = W^S \cup w_n^S$ 
run QPP using waypoints in  $W^S$ 

```

Formally, the segments of the Voronoi diagram are defined by all points that are equidistant to the two nearest landmarks. The dual graph for a Voronoi diagram corresponds to the Delaunay triangulation, which connects triplets of landmarks $\mathcal{L}_{ijk} = \{\ell_i^S, \ell_j^S, \ell_k^S\}$ such that no other landmarks are located inside the circumcircle of the triangle $\text{tri}(\mathcal{L}_{ijk})$. Waypoints located along the segments of the Delaunay triangulation represent locations at which the path passes *between* or *past* landmarks, and waypoints along the Voronoi diagram are locations at which the path moves *away* from one landmark *towards* another. These points, in addition to waypoints specifically designated by the human, define the set of sketched waypoints $w_i^S \in W^S$ used by the QPP. This waypoint extraction procedure is shown in Algorithm 1. Unless otherwise specified, all examples given in this work employ this waypoint extraction routine.

As an example of how the QPP performs using the waypoints obtained from

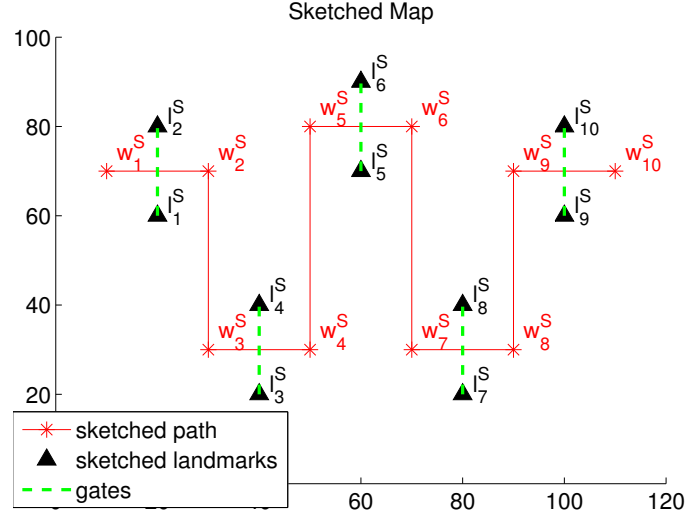


Figure 3.5: A sketch map consisting of landmarks (triangles) and waypoints (*). The desired path passes between five landmark pairs, or 'gates'.

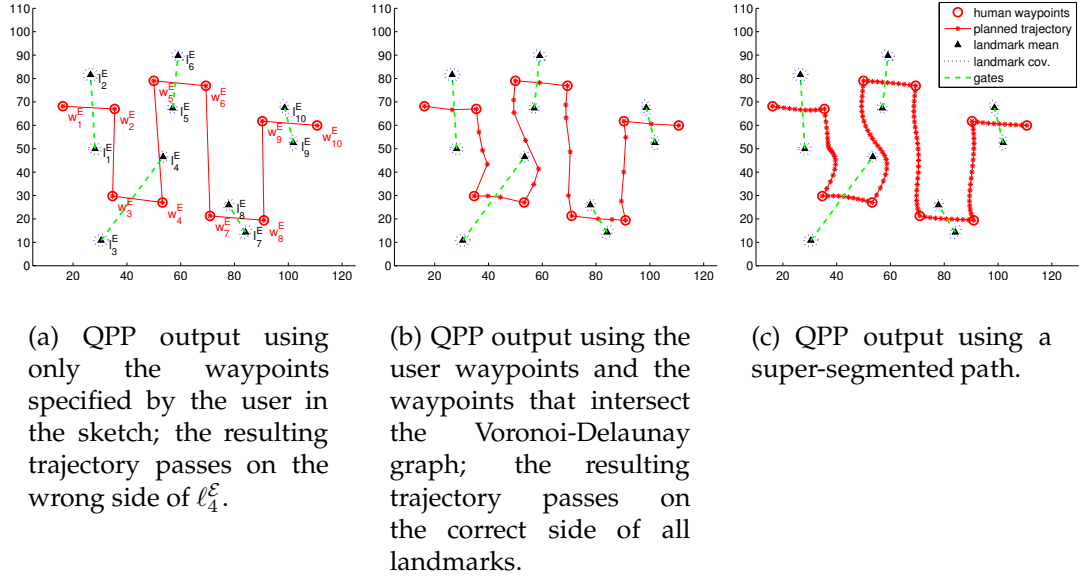


Figure 3.6: The planned paths obtained from the QPP using: (a) sparse waypoints defined by user, (b) the path with waypoint augmentation using the Voronoi-Delaunay graph, and (c) a super-segmented path.

the Voronoi-Delaunay graph, consider Figures 3.5 and 3.6, which illustrate the effects of different path segmentation strategies. Figure 3.5 shows the sketched map (path and landmarks), where the desired path (defined as a sequence of waypoints $w_{1:n}^S$) passes between five pairs of landmarks (referred to as ‘gates’). Figure 3.6(a) shows the output of the QPP on the estimated environment using just these initial waypoints. In this case, the waypoints provided by the human are sparse, and although the robot still successfully navigates most of the desired path, the planned trajectory fails to go around the right of landmark ℓ_4^E . This is because there is not enough information about *how* the robot should get from waypoint w_4^E to waypoint w_5^E . The second case, shown in Figure 3.6(b), uses the initial sparse waypoints (from Fig. 3.5) as well as the output of the Voronoi-Delaunay point extraction (Algorithm 1). By adding these extracted waypoints along the path, the QPP is able to plan a better trajectory to the right of landmark ℓ_4^E . The QPP can also be applied to a ‘super-segmented’ path, in which arbitrarily-close waypoints are extracted along the sketched path, as shown in 3.6(c). One can argue that the super-segmented path represents the complete qualitative trajectory as intended by the human. Comparing Figure 3.6(b) and (c), the QPP trajectory produced from the Voronoi-Delaunay segmentation is nearly identical to that of the super-segmented path, and therefore provides a good approximation of the path intended by the human using fewer waypoints and less computation.

In addition to planning, the waypoints created by intersecting the Voronoi-Delaunay graph are also convenient ‘checkpoints’ as the robot traverses its trajectory, as they correspond to locations with unique relationships to the immediately surrounding landmarks. If the robot reaches a waypoint and the observed environment is very different than expected (for example, the waypoint is de-

finer by the Voronoi-Delaunay graph as being *between* two landmarks, but one of the landmarks has not yet been observed), this may be an indication that: a) the robot is lost, b) the sketched map is extremely distorted, c) the sensors are not able to sufficiently observe the environment, and/or d) the transformation between the environment and the sketched map has been incorrectly estimated (presented in Section 3.2.3). Any of these situations may warrant corrective measures; for example, the robot may decide to query the human user for help.

3.2.3 Estimation of Sketch Transformation

The equations in Section 3.2.1 assume that the coordinates of the sketched map have been scaled and rotated appropriately with respect to the true environment. However, in many cases the transformation between the sketched map and the true environment is not specified *a priori*, and must be inferred from robot observations. This is analogous to a human attempting to navigate around a city using a map with no bar scale or compass; by observing the locations of several buildings or other landmarks, a human could infer how the map is scaled, oriented, and translated with respect to the environment. In the same way, the affine transformation matrix $T^{\mathcal{E} \rightarrow \mathcal{S}}$ is estimated using the current estimate of the environment:

$$X^{\mathcal{E}} = \begin{bmatrix} \bar{\ell}_{1,x}^{\mathcal{E}} & \bar{\ell}_{1,y}^{\mathcal{E}} & 1 \\ \vdots & \vdots & \vdots \\ \bar{\ell}_{m,x}^{\mathcal{E}} & \bar{\ell}_{m,y}^{\mathcal{E}} & 1 \end{bmatrix}_{m \times 3}$$

and landmark locations as given in the sketched map:

$$X^S = \begin{bmatrix} \ell_{1,x}^S & \ell_{1,y}^S & 1 \\ \vdots & \vdots & \vdots \\ \ell_{m,x}^S & \ell_{m,y}^S & 1 \end{bmatrix}_{m \times 3}$$

Using weighted least squares, the 3×3 affine transformation matrix is calculated as:

$$T^{\mathcal{E} \rightarrow \mathcal{S}} = \left((X^{\mathcal{E}})^T W X^{\mathcal{E}} \right)^{-1} (X^{\mathcal{E}})^T W X^S \quad (3.6)$$

where W is a diagonal weighting matrix derived from the current landmark location covariances $\Sigma_j^{\mathcal{E}}(t)$:

$$W = \begin{bmatrix} |\Sigma_1^{\mathcal{E}}(t)| & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & |\Sigma_m^{\mathcal{E}}(t)| \end{bmatrix}_{m \times m}^{-1}$$

The weighting matrix W is defined such that landmarks which have better estimates are considered more strongly in the transformation estimate than landmarks whose locations are less certain.

The sketched landmarks ℓ_j^S and waypoint locations w_i^S are then transformed into environment coordinates at each time step using

$$\begin{bmatrix} \hat{w}_{i,x}^S & \hat{w}_{i,y}^S & 1 \end{bmatrix} = \begin{bmatrix} w_{i,x}^S & w_{i,y}^S & 1 \end{bmatrix} (T^{\mathcal{E} \rightarrow \mathcal{S}})^{-1}$$

$$\begin{bmatrix} \hat{\ell}_{j,x}^S & \hat{\ell}_{j,y}^S & 1 \end{bmatrix} = \begin{bmatrix} \ell_{j,x}^S & \ell_{j,y}^S & 1 \end{bmatrix} (T^{\mathcal{E} \rightarrow \mathcal{S}})^{-1}$$

These transformed landmarks and waypoints are then used in the QPP described in Section 3.2.1. Note that a sufficient number of landmarks must be

observed in order to calculate this transformation properly. In two dimensions, at least three landmarks from the sketched map must be observed in the environment, whose x/y locations are linearly independent. Because of uncertainties in sketch locations, landmark estimates and data association, additional landmarks generally leads to better results.

If the orientation of the sketched map is known (for example, North is indicated), the scaling and translation can be estimated in a similar fashion:

$$\hat{T}^{\mathcal{E} \rightarrow \mathcal{S}} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ t_x & t_y & 1 \end{bmatrix} \quad (3.7)$$

where s_x and t_x are defined as:

$$\begin{bmatrix} s_x & 0 \\ t_x & 1 \end{bmatrix} = \left[(X_x^{\mathcal{E}})^T W X_x^{\mathcal{E}} \right]^{-1} \left[(X_x^{\mathcal{E}})^T W X_x^{\mathcal{S}} \right]$$

$$X_x^{\mathcal{E}} = \begin{bmatrix} \bar{\ell}_{1,x}^{\mathcal{E}} & 1 \\ \vdots & \vdots \\ \bar{\ell}_{m,x}^{\mathcal{E}} & 1 \end{bmatrix}_{m \times 2}, \quad X_x^{\mathcal{S}} = \begin{bmatrix} \ell_{1,x}^{\mathcal{S}} & 1 \\ \vdots & \vdots \\ \ell_{m,x}^{\mathcal{S}} & 1 \end{bmatrix}_{m \times 2}$$

and similarly for s_y and t_y . Given that this requires solving for fewer unknowns (it is a simplified case of Equation 3.6), results when orientation is known are generally better.

3.2.4 Extensions Addressing Implementation Issues

The QPP algorithm described in Sections 3.2.1–3.2.3 makes several assumptions about the sketched map and the environment in which the robot is operating.

First, it is assumed that the robot observes enough of the environment to calculate $T^{\mathcal{E} \rightarrow \mathcal{S}}$, and subsequently $w_i^{\mathcal{E}}$. This requires the robot sensors to have an appropriate range/resolution to identify landmarks, and for landmarks to not be obscured from view. Secondly, the output of the QPP presumes that the closed-form solution $w_i^{\mathcal{E}}$ is a reachable point, i.e., the waypoint is not inside an obstacle or at an otherwise inaccessible location. Compensation for these practical implementation issues is discussed in the following sub-sections.

Sensor Constraints and Landmark Sparseness

The ability to perform navigation using landmarks is naturally subject to the sensor constraints of the robot. For example, in the case of a range-limited sensor such as a laser scanner, the performance of the QPP is a direct function of the sensor range, as related to the size and sparseness of the environment. The notion of a landmark *effective distance* (ED) is introduced as a useful measure of sensor range. The landmark ED is defined as the average distance between true landmark pairs connected by the Delaunay graph:

$$\text{ED} = \frac{1}{s} \sum_{n=1}^s \|\ell_{D_n(1)}^T - \ell_{D_n(2)}^T\| \quad (3.8)$$

where $\ell_{D_n(1)}^T$ and $\ell_{D_n(2)}^T$ are the true locations of the two landmarks connected by the n^{th} Delaunay segment, and s is the total number of Delaunay segments in the graph. Recall from Section 3.2.2 that the Delaunay triangulation connects triplets of neighboring landmarks; therefore ED is a measure of the average distance between the ‘closest’ landmarks in the true environment. If landmarks are densely clustered together with respect to the sensor range ($r \gg 1\text{ED}$), the robot should have several landmarks within its sensor field of view most of the

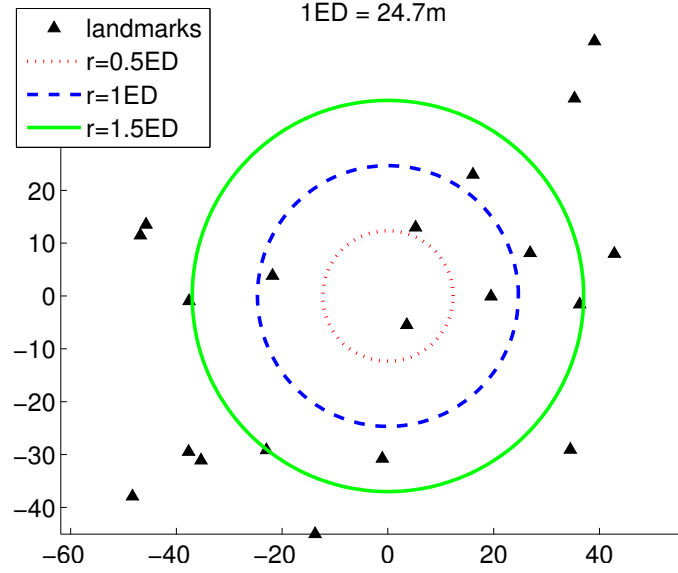


Figure 3.7: Three sensor fields of view shown as a function of the landmark effective distance (ED) in a randomly-generated environment of 20 landmarks. In this example, the effective distance is 24.7m.

time. On the other hand, if landmarks are sparse and located very far apart and/or the sensor range is small ($r \ll 1ED$), the robot may travel for extended periods of time without any landmarks in its field of view, and may be more prone to getting lost.

Figure 3.7 illustrates this point for a field with twenty randomly-placed landmarks. For a small sensor radius ($r = 0.5ED$), the robot sensors can observe only 0–3 landmarks as it navigates the environment, whereas a large radius ($r = 1.5ED$) enables the observation of more than half of all landmarks in the environment at any one time. For the remainder of this work, effective distance is used as a measure for characterizing the robot sensor range as well as the amount of sketch map distortion. Section 3.3.2 examines how these properties affect the performance of the QPP.

Obstacle Avoidance

In many cases, low-level obstacle avoidance can be implemented to help the robot safely traverse between planned waypoints along the trajectory. For this to be a successful obstacle-avoidance strategy however, the waypoints themselves must not be located inside obstacles. If there exists a known area in which a waypoint must (or must not) be located, the QPP can be modified to add such constraints. Since the QPP is already formulated as a quadratic optimization problem (Eqn. 3.3), many types of environmental constraints can naturally be accommodated.

For example, consider the case where each landmark is an obstacle, and the robot must avoid entering the 2σ uncertainty ellipse of each obstacle. Defining the parameter set for a 2D ellipse $[x_c, y_c, a, b, \theta]$, where (x_c, y_c) is the center point location, a is the semimajor axis, b is the semiminor axis, and θ is the angle between a and the global x-axis, a point (x, y) is located outside an ellipse if it obeys the inequality [74]:

$$\begin{aligned} Ax^2 + Bxy + Cy^2 - (2Ax_c + By_c)x \\ - (2Cy_c + Bx_c)y + (Ax_c^2 + Bx_cy_c + Cy_c^2) > 1 \end{aligned} \quad (3.9)$$

where:

$$\begin{aligned} A &= \left(\frac{\cos^2 \theta}{a^2} + \frac{\sin^2 \theta}{b^2} \right) \\ B &= -2 \cos \theta \sin \theta \left(\frac{1}{a^2} - \frac{1}{b^2} \right) \\ C &= \left(\frac{\sin^2 \theta}{a^2} + \frac{\cos^2 \theta}{b^2} \right) \end{aligned}$$

Equation 3.9 can be added as a constraint to the optimization problem minimizing Equation 3.2. This is a quadratically constrained quadratic program, and

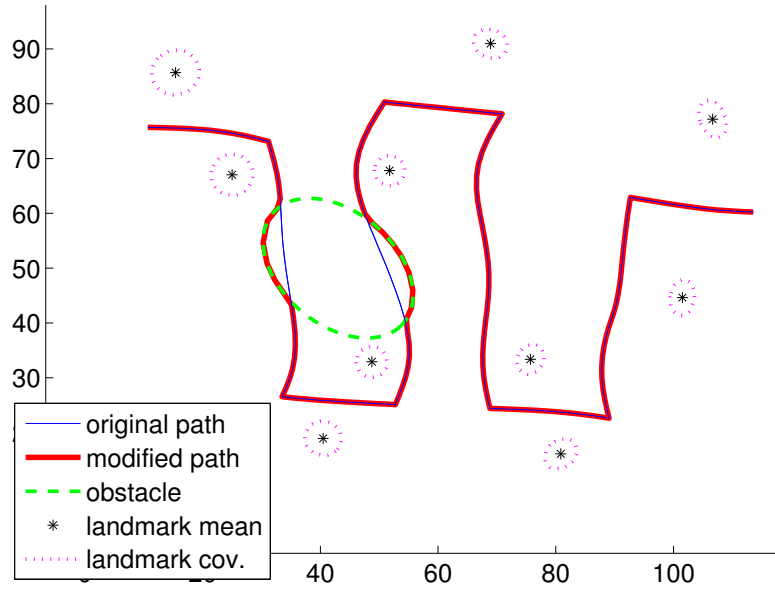


Figure 3.8: The modified QPP is able to plan around an obstacle by incorporating an elliptical constraint (Equation 3.9) in the waypoint optimization.

while the optimization can no longer be solved in closed form, it can be solved efficiently using interior point methods [75]. Figure 3.8 shows an example of a planned path using QPP with an elliptical obstacle. The closed-form QPP without constraints would plan a trajectory through the ellipse (blue line), but the modified QPP that minimizes Equation 3.3 *subject to* Equation 3.9 successfully avoids the obstacle (red line). The QPP lends itself naturally to the addition of such environmental constraints, however care should be taken to avoid waypoint ‘chattering’ (i.e., switching between local minima) if convexity no longer holds.

3.3 Experimental Results

The QPP was simulated in several scenarios to study its sensitivity to sketched map inaccuracies, limited sensor capabilities, landmark sparseness, and different human mapping styles. Section 3.3.1 presents a few illustrative examples to emphasize advantages of the QPP over other approaches. In Section 3.3.2, a Monte Carlo analysis is described comparing the performance of a naïve planner to that of two variations of the QPP for different map distortions and sensor ranges. Section 3.3.3 presents a set of human trials, conducted to examine the effectiveness of the QPP for a navigation task over multiple users.

3.3.1 Illustrative Examples

In this section, two illustrative examples are presented to highlight some of the features of the QPP and distinguish its qualities from previous approaches. A differential drive robot is simulated, with robot pose (x, y, θ) and landmark locations $\ell_{(\cdot)}^{\mathcal{E}}$ estimated using EKF-SLAM [76] in a $100\text{m} \times 100\text{m}$ environment. The robot is equipped with a LIDAR sensor, with a maximum sensor range of $r_{\max} = 1.5\text{ED}$ with Gaussian noise $\sigma_r = 0.05\text{ED}$, and bearing ranges from $\theta \in [-3\pi/4, 3\pi/4]$ with Gaussian noise $\sigma_b = 0.01$ radians.

Figures 3.9 and 3.10 illustrate the performance of the QPP in two challenging scenarios. In Figure 3.9(left), the human has drawn a straight-line path between four pairs of landmarks. Figure 3.9(right) shows the same straight-line path with respect to the true environment (dashed line). Because of the true landmark locations, it is clear that the original sketched path would not

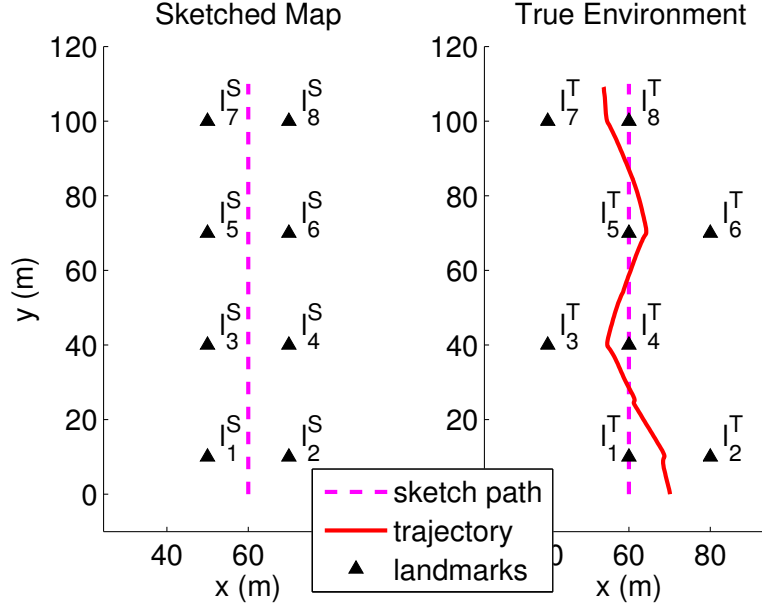


Figure 3.9: The sketched map (left) includes eight landmarks, with a path passing between four pairs in a straight line. In the true environment (right), the landmark locations require the robot trajectory to weave between landmark pairs in order to maintain appropriate spatial relationships. (sensor range = 46m or 1.5ED)

maintain the desired spatial relationships to the landmarks. Many of the previous approaches described in Section 1.2 might experience difficulties in this scenario, especially [22] and [24], which pre-compute robot commands at key points along the path. By contrast, the QPP does not define the shape of the path *a priori*, allowing it to modify the trajectory to maintain appropriate relative spatial relationships to all the landmarks. The solid line in Figure 3.9(right) shows the output of the QPP, which is able to plan a trajectory that successfully passes between the desired landmark pairs.

Another advantage of the QPP is its robustness to missing landmarks. Recall from Equation 3.1 that a landmark spring constant k_i^j defines the degree to which landmark j influences the location of waypoint i , and is inversely propor-

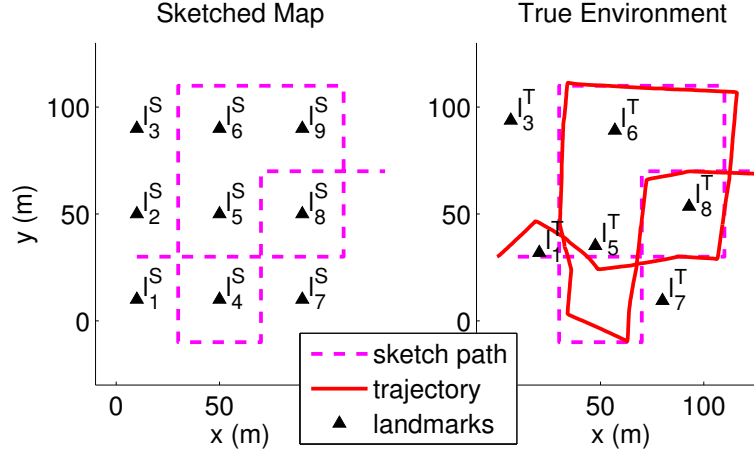


Figure 3.10: The sketched map (left) shows a path that passes around and between nine landmarks. In the true environment (right), three landmarks are not present, due to their not existing or not being sensed by the robot. (sensor range = 78m or 1.5ED)

tional to $|\Sigma_j^{\mathcal{E}}(t)|^{1/2}$. For an unobserved landmark, the error covariance tends to infinity, and has no influence on the resulting trajectory; similarly, landmark locations that are observable but very uncertain, are weighted appropriately. Figure 3.10 shows an example where three of the nine landmarks in the sketched map (left) are not present in the true environment (right). Figure 3.10(right) shows the output path of the QPP. Despite the missing landmarks, the QPP is still able to plan a reasonable trajectory using the remaining observed landmarks (for example, the trajectory passes above ℓ_1^T and below ℓ_5^T , even though landmarks ℓ_2^T and ℓ_4^T are not observed). Methods such as those in [34], [35] and [24], which perform navigation by matching states in the real world to those expected from the sketch map, may cause the robot to become lost if one or more matches are missed or made in error. The QPP is robust to missing landmarks because all previously observed landmarks are used for planning.

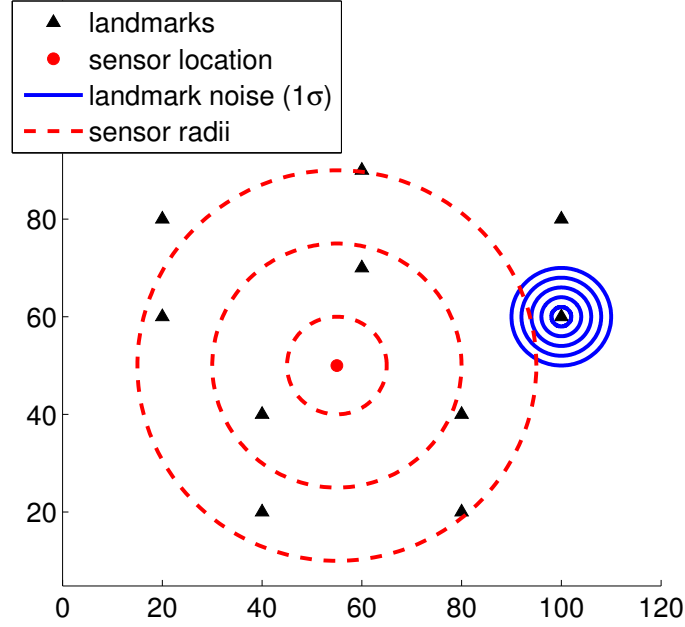


Figure 3.11: The environment used for the sensitivity analysis. The sensor radius was varied $r \in [0.3, 0.75, 1.2ED]$, and Gaussian noise was added to each landmark location $\sim \mathcal{N}(0, \sigma^2)$ where $\sigma \in [0.06, 0.12, 0.18, 0.24, 0.3ED]$.

3.3.2 Sensitivity to Sensor Limitations and Map Inaccuracies

A key strength of the QPP is robustness to map inaccuracies and incomplete knowledge of the true environment. To illustrate this, a sensitivity analysis was performed on the example sketch in Figure 3.5. In this scenario, the human has placed ten landmarks in a 100×100 map and drawn a path that passes between pairs of landmarks, or ‘gates’. The sensor radius is varied over three values $r \in [10, 25, 40\text{m}]$ ($0.3, 0.75, 1.2ED$). Intuitively, a robot with sensor radius $r = 0.3ED$ has a very small sensor range with respect to the sparsity of landmarks in the environment, and frequently has no landmarks within its field of view. On the other hand, a robot with a sensor radius of $r = 1.2ED$ is able to observe much more of the environment, with an average of seven landmarks at any time.

To simulate map inaccuracies, the true environment was generated by adding Gaussian noise to each landmark x and y locations, $\mathcal{N}(0, \sigma^2)$, where $\sigma \in [2, 4, 6, 8, 10\text{m}]$ (0.06, 0.12, 0.18, 0.24, 0.3ED). In order to visualize how these parameters affect the simulations, three robot sensor radii and five landmark noises (1σ ellipses) used in the sensitivity studies are illustrated in Figure 3.11. Note that when $\sigma = 0.3\text{ED}$, the sketched map is significantly different from the true environment, such that neighboring landmarks may even ‘switch’ their relative positions. One might expect this type of situation to cause confusion even to a human navigator, who may become disoriented if landmarks are observed to have switched relative locations in the map.

Three planners are evaluated for comparison:

1. Simple Planner (SP): estimates the full map transformation $T^{\mathcal{E} \rightarrow \mathcal{S}}$ (Equation 3.6), and projects the sketched waypoints onto the estimated environment via: $w_i^{\mathcal{E}} = w_i^{\mathcal{S}} (T^{\mathcal{E} \rightarrow \mathcal{S}})^{-1}$.
2. Qualitative Path Planner (QPP): estimates the full map transformation $T^{\mathcal{E} \rightarrow \mathcal{S}}$ (Equation 3.6), and optimizes waypoint locations as in Section 3.2.
3. Qualitative Path Planner with Known Orientation (QPPK): estimates map scaling and translation only $\hat{T}^{\mathcal{E} \rightarrow \mathcal{S}}$ (Equation 3.7), and optimizes waypoint locations as in Section 3.2.

All three planners use waypoint locations extracted using the Voronoi-Delaunay graph described in Section 3.2.2. A total of 50 Monte Carlo simulations were run for each combination of σ and r , for each of the three planners.

The ‘goodness’ of a planned trajectory from a qualitative map is a somewhat ambiguous characteristic to quantify, so the gate scenario in Figure 3.5 was de-

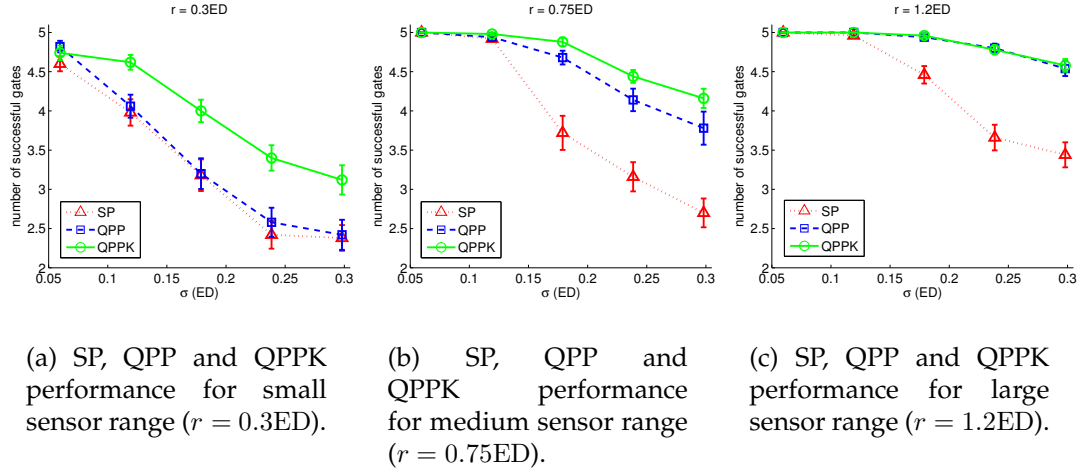


Figure 3.12: Results of a sensitivity study for the map in Figure 3.5. Each data point represents the mean and standard error of 50 Monte Carlo simulations. The number of successful landmark gates is plotted as a function of the landmark distortions σ for (a) small, (b) medium, and (c) large sensor ranges, r .

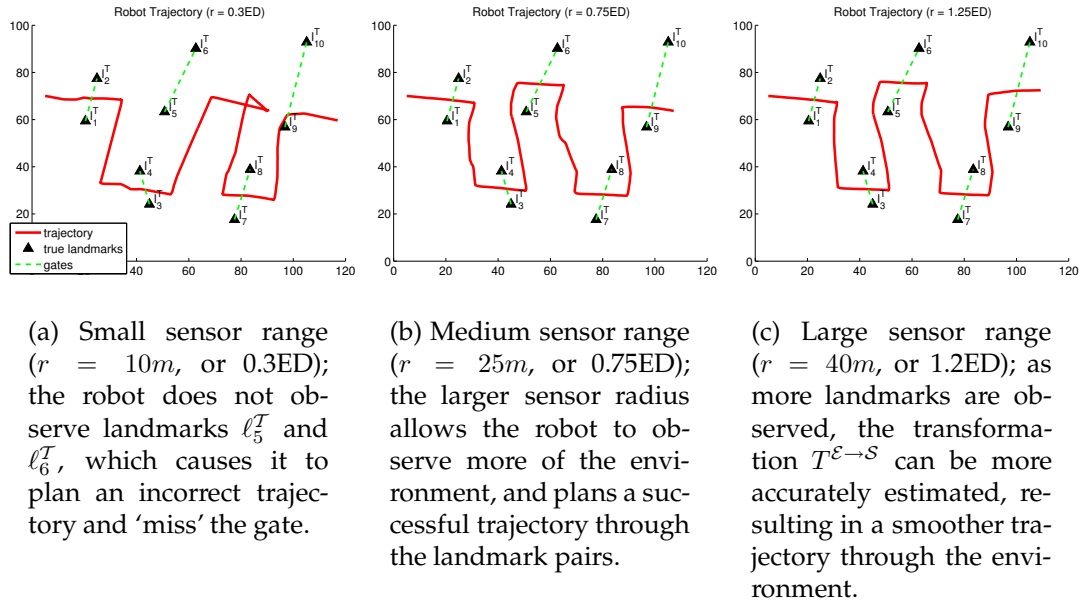


Figure 3.13: The QPP trajectory for the same sketch and true environment ($\sigma = 6\text{m}$, or 0.18ED) for three different sensor radii.

vised in order to objectively evaluate planner ‘success’. As a performance metric, the number of successful landmark gates that the robot passes through is one measure of how well the robot is able to follow the path *intended* by the human. A gate is considered successfully traversed if the robot passed through the landmark pair exactly once and in the correct direction. Figure 3.12 shows the average number of successful gates over 50 Monte Carlo simulations for (a) small, (b) medium, and (c) large sensor ranges with varying degrees of map inaccuracies, σ . For all three planners, the robot is most successful at traversing the gates when σ is small and r is large. These results are not surprising, as a small σ implies that the sketched map is a close approximation to the true environment, and a large r allows the robot to observe more of the environment, thereby providing more information for estimating an accurate map transformation $T^{\mathcal{E} \rightarrow \mathcal{S}}$.

For the small sensor range case shown in Figure 3.12(a), the performance of both the SP and QPP drop quickly as σ increases and the sketch map deviates further from the true environment. Both of these planners perform consistently worse than the QPPK, indicating that significant error is introduced by poorly estimating $T^{\mathcal{E} \rightarrow \mathcal{S}}$. However, even the QPPK misses approximately 40% of landmark gates for $\sigma = 0.3ED$, due to its inability to plan around unobserved landmarks. For the large sensor range case shown in Figure 3.12(c), the QPP and QPPK perform equally well because the robot is able to observe enough landmarks to accurately estimate the full map transformation $T^{\mathcal{E} \rightarrow \mathcal{S}}$. Even for large σ , both the QPP and QPPK are able to successfully traverse an average of over 4.5 gates because the local environment about each waypoint is appropriately considered through the $(d_i^j)^p$ term in k_i^j . By contrast, the SP performance degrades significantly as σ increases; even though the SP is also able to accurately

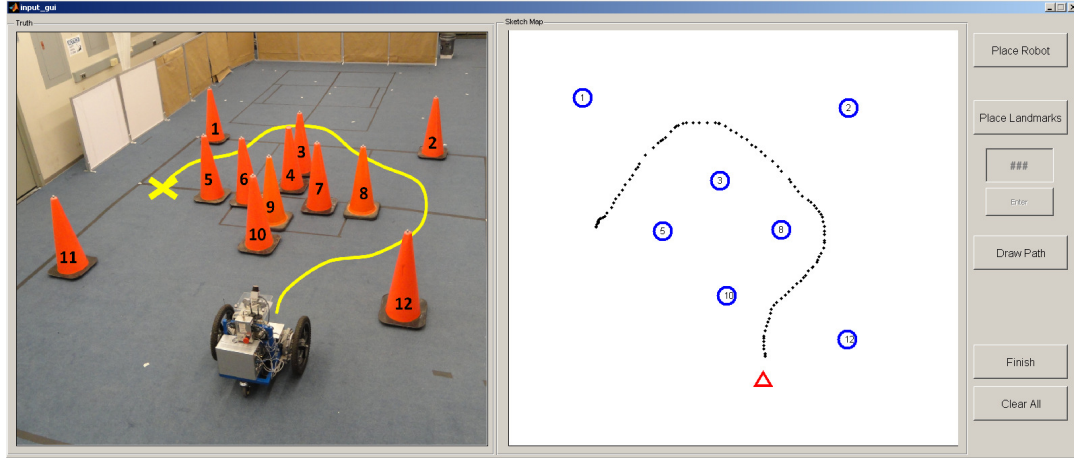


Figure 3.14: The interface used for collecting sketch maps in human trials. On the left, an image is shown of the environment consisting of a robot, twelve landmarks (numbered cones), a goal position (X), and a desired path (for sets `PathAny` and `PathAll` only). On the right, the user sketches a map to illustrate how the robot should navigate to the goal.

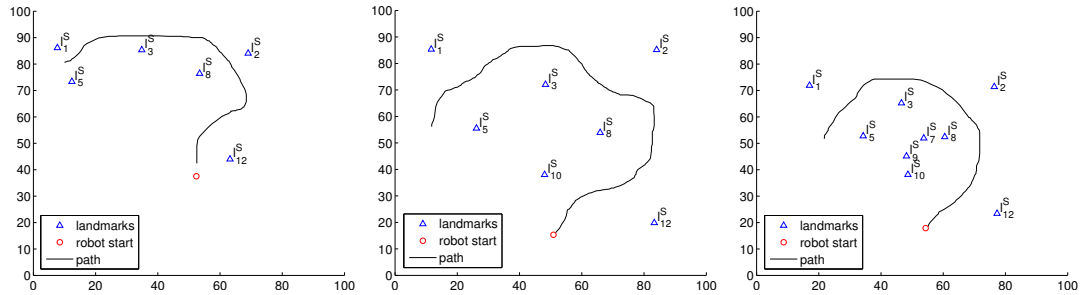


Figure 3.15: Three users' sketch maps for the `PathAny` set (Map 4, see Figure 3.14(left)). Each user indicates only the landmarks s /he thinks are necessary for navigation, and draws a path to the goal.

estimate $T^{\mathcal{E} \rightarrow \mathcal{S}}$, it doesn't compensate for local deviations in landmark locations, causing it to miss more gates.

Figure 3.13 presents an illustrative example of how the robot sensor radius affects the quality of the trajectory planned by the QPP for the same sketch with

medium map distortion ($\sigma = 0.18ED$). Figure 3.13(a) shows the small sensor range case ($r = 0.3ED$); the robot misses the gate between landmarks ℓ_5^T and ℓ_6^T due to its inability to observe the landmarks. Figure 3.13(b) shows the medium sensor range case ($r = 0.75ED$); the robot successfully traverses all five gates, but the trajectory is slightly jagged due to the frequent recalculation of the map transformation $T^{\mathcal{E} \rightarrow \mathcal{S}}$ as new landmarks are observed. Finally, Figure 3.13(c) shows the large sensor radius case ($r = 1.2ED$); the robot is able to observe most of the landmarks at any one time, and the QPP produces a trajectory that passes through all five gates successfully, while also producing a smoother path. These results, and those presented with the discussion of Figure 3.12, illustrate that the QPP is less sensitive to large distortions in the sketched map, especially as the sensor radius is larger and multiple landmarks are observed at any one time.

3.3.3 Human Trial Experiments with the QPP

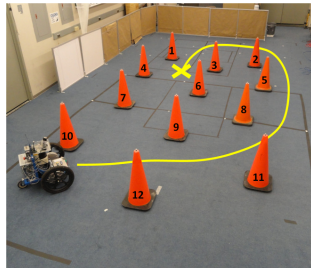
A set of human trials was conducted to study the efficacy of the QPP for a real robot navigation task. Ten users provided four sets of sketched maps using the interface shown in Figure 3.14. Each set consisted of five different environments containing of 12 landmarks (numbered traffic cones), each displayed on the left side of the interface. On the right side, users sketched a map to guide the robot to the specified goal location (X). The sketched map consisted of the robot starting location, labeled landmarks (where the classes corresponded to the cone numbers in the left image), and the desired path for the robot to take. After each user sketched the map, a simulated robot planned a trajectory in its observed environment using the QPP (low-level collision avoidance was not used in these simulations). This hybrid approach (human experimental trials with

post-simulation) allows for subsequent analysis of performance as a function of sensor range. The robot pose and landmark location estimation was performed using UKF-SLAM [77], and landmark observations were simulated as LIDAR-like measurements, with Gaussian range noise $\sigma_r = 0.03\text{ED}$ and bearing noise $\sigma_b = 0.02\text{rad}$.

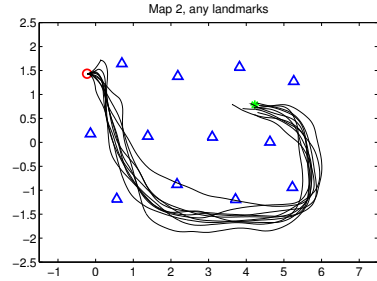
A 2×2 test matrix (corresponding to path specification and number of sketched landmarks) was used to investigate the QPP performance across different users, as well as how humans performed the mapping task (e.g., landmark selection and path drawing). The four sets of tests vary the number of landmarks to sketch (any vs. all landmarks) and path-to-goal (any vs. follow suggested path). For clarity, these are defined as follows:

1. `FreeAny`: Place as many (or few) landmarks on the map as necessary and send the robot to the goal via any desired path.
2. `FreeAll`: Place all landmarks on the map and send the robot to the goal via any desired path.
3. `PathAny`: Place as many (or few) landmarks on the map as necessary and send the robot to the goal via the path indicated.
4. `PathAll`: Place all landmarks on the map and send the robot to the goal via the path indicated.

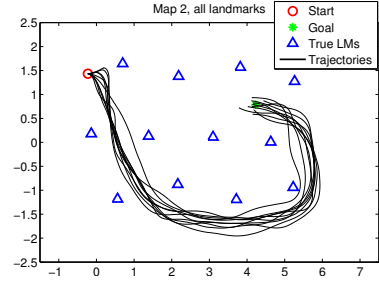
Users were not given any instructions as to the scaling or orientation of the true environment, and were told only that the robot was equipped with a sensor that could locate and identify the landmarks. For sets `FreeAny` and `PathAny`, in which users could choose which landmarks to include in their sketched map, users were told to provide as much information as they deemed necessary for



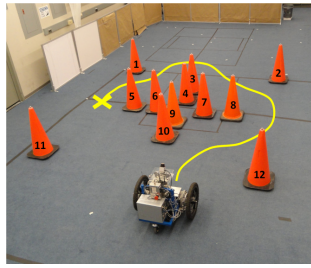
(a) Photo of Map 2 environment



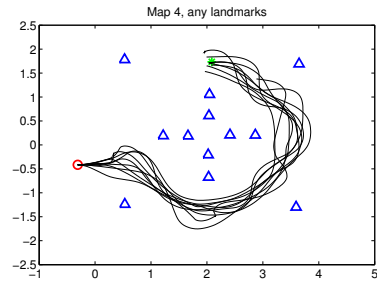
(b) Map 2 PathAny; average 9.3 landmarks in sketch



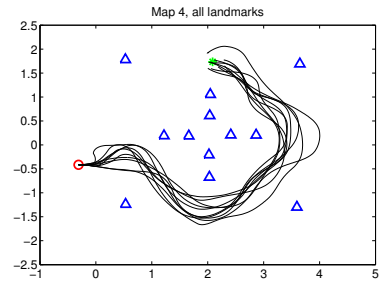
(c) Map 2 PathAll



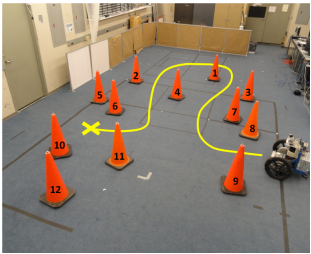
(d) Photo of Map 4 environment



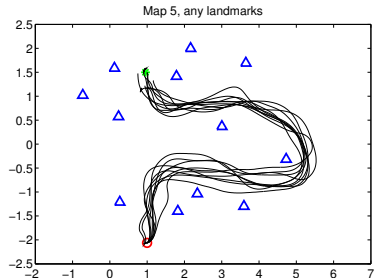
(e) Map 4 PathAny; average 7.8 landmarks in sketch



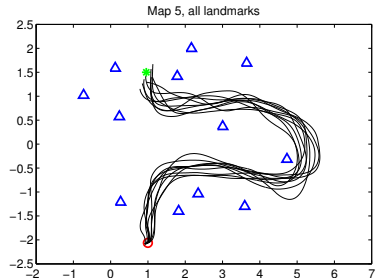
(f) Map 4 PathAll



(g) Photo of Map 5 environment



(h) Map 5 PathAny; average 9.0 landmarks in sketch



(i) Map 5 PathAll

Figure 3.16: Trajectories generated by QPP in the true environment based on sketched maps provided by ten human users for $r = 1.5ED$.

a ‘reasonable human’ to perform the navigation task. Figure 3.15 illustrates a sample of the wide variety of sketched maps provided by human users for the PathAny case (Map 4, as shown in Figure 3.14(left)). Notice that each user

Table 3.1: Mean (Standard Error) statistics from the human trials.

	FreeAny	FreeAll	PathAny	PathAll
final goal offset (m), $r = 1.5ED$	0.203 (0.017)	0.190 (0.018)	0.209 (0.023)	0.177 (0.014)
average landmark offset (m)	0.184 (0.011)	0.221 (0.012)	0.251 (0.013)	0.207 (0.010)
# sketched landmarks	7.060 (0.275)	12 (0)	8.820 (0.249)	12 (0)

chose different landmarks to include in the sketched map, and the shapes of the sketched paths vary greatly.

Figure 3.16 shows a sampling of the resulting trajectories for $r = 1.5ED$ for `PathAny` and `PathAll`. A few observations can be made from these results. First, the units of the sketched maps were arbitrary (100×100), and most users sketched the map as they viewed it in the photo, which was approximately 90° rotated from the true environment. Second, the trajectories in the `PathAny` set were slightly less uniform across users and were ‘worse’ (with respect to closeness and relative passing) at following the indicated path. This can be seen particularly in Figure 3.16(b), where one trajectory passes over two landmarks; this was caused by the user excluding these landmarks from the sketched map, so the QPP ignored them when planning. By contrast, trajectories in the `PathAll` set were much more uniform and maintained the appropriate spatial relationships to the landmarks along the route. By including more landmarks in the sketched map, the QPP was better able to estimate the map transformation $T^{\mathcal{E} \rightarrow \mathcal{S}}$. Also, by forcing users to sketch all landmarks (whether or not they seemed ‘important’ for the navigation task), users were not able to overlook landmarks, eliminating landmark closeness problems such as the one seen in Figure 3.16(b).

The final goal offset (defined as the distance from the final position of the

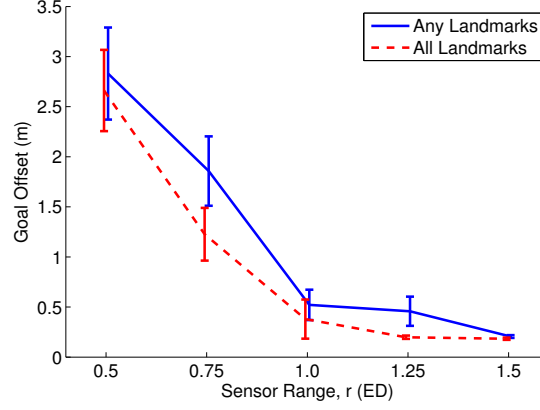


Figure 3.17: Average final goal offset as a function of sensor range r for the case where all landmarks were included in the sketch map (FreeAll and PathAll) and the case where the user chose which landmarks to sketch (FreeAny and PathAny).

robot to the desired goal point) is an intuitive metric for determining the success of each planned trajectory. Figure 3.17 shows the final goal offset for different sensor ranges. As expected, the QPP performed poorly when the sensor range was small ($r < 1\text{ED}$) because the robot was often not able to observe enough landmarks to localize itself in the true environment. Note that for 3 of the 5 maps, the robot starting position was at a distance greater than 0.9ED from the nearest landmark in the environment. Therefore, for $r < 0.9\text{ED}$, no landmarks were visible from the robot start position, and the QPP was unable to estimate $T^{\mathcal{E} \rightarrow \mathcal{S}}$. In some of these cases, the robot traveled in the opposite direction from the landmarks and was never able to recover. For $r \geq 1\text{ED}$ however, at least one landmark was within the initial sensor range. As long as this closest landmark was included in the sketch map (mandatory for FreeAll and PathAll), the QPP was able to successfully estimate and navigate in the environment.

Table 3.1 presents the statistics for the human trials for a large sensor radius, $r = 1.5\text{ED}$, for all four test cases. The average final goal offset was smaller for the

cases where all landmarks were included in the sketch map (0.18m, or 0.12ED) compared to those where the users chose the landmarks to include in the sketch (0.21m, or 0.14ED). This difference, while not statistically significant, suggests that the QPP performs better when more landmarks are present, as expected, and that human users are able to draw more accurate paths when required to include all landmarks.

On average, users sketched 7-9 landmarks (58–75% of total) in the map for the `FreeAny` and `PathAny` cases, indicating that users prefer to select a subset of landmarks that they believe are vital to the navigation task, rather than over-specifying the environment. Since the QPP does not require or assume all landmarks in the environment have a corresponding sketch landmark, it is able to accommodate this user preference. The average landmark offset corresponds to the overall distortion in the sketch map, and was found by transforming the sketch map into the coordinates of the true environment using the transformation $T^{\mathcal{E} \rightarrow \mathcal{S}}$ and calculating the average distance between the true landmarks and corresponding sketched landmarks. The average landmark offset was approximately 0.18–0.25m (0.12–0.17ED); as discussed in Section 3.3.2, this amount of sketch map distortion is within reasonable range to achieve good QPP performance for large sensor ranges. Interestingly, although the average landmark offset was larger for `FreeAll` cases than `FreeAny`, the average final goal offset was still smaller for `FreeAll` cases than `FreeAny`, further suggesting that the QPP performs better when there are more landmarks in the sketch map, even if the map is more distorted.

The results of these human trials illustrate the effectiveness of the QPP for real robot navigation tasks. With a sufficient sensor range, the robot was able

to successfully navigate its environment using sketched maps from novice human users. Despite widely varying sketch inputs, such as those shown in Figure 3.15, the resulting trajectories were relatively homogeneous (Fig. 3.16). Because the QPP mimics how humans naturally communicate route instructions, users were able to successfully interact with the system with negligible training. These results indicate that the QPP is a useful, intuitive, and effective tool for commanding a desired path to a robotic agent, without the need for an accurate truth map.

3.4 Chapter Conclusions

This chapter introduced the Qualitative Path Planner, a method for controlling a mobile robot using only an approximate human-provided map and measurements to identifiable landmarks. Waypoint planning is posed as a quadratic optimization problem in order to maintain appropriate spatial relationships to landmarks in the environment. This optimization yields a closed-form solution when no additional constraints are added, but can be naturally modified to include constraints (e.g., for obstacle avoidance).

Waypoints are extracted by using the intersection points between the sketched path and the Voronoi-Delaunay graph created by sketch landmarks. This method mimics how humans navigate by creating waypoints at locations where the path passes *between* or *past* landmarks, and locations where the path moves *away* from one landmark *towards* another. These waypoints help preserve the desired shape of the trajectory without arbitrarily segmenting the path, and can serve as natural ‘checkpoints’ along the route at which the robot can query

the human if the observed environment is too dissimilar from the sketch.

Unlike other approaches, the QPP does not assume straight-line path segments, and does not rely on sensor ‘template matching’ to trigger a change in the planned route. Results of a sensitivity study show that the QPP is robust to inaccuracies in the sketch map and sensor range. For small sensor ranges relative to the sparseness of the landmarks, not enough landmarks are observed to enable successful estimation of an accurate map transformation, causing the robot to get ‘lost.’ This behavior is similar for human navigators when landmarks are too sparse. For large sensor ranges relative to the sparseness of the landmarks, however, the QPP is able to successfully navigate the environment, even when the sketched map is quite distorted compared to the truth.

Human trials were performed to demonstrate the robustness of the QPP to different users’ sketched maps and its efficacy for mobile robot control. With sufficient sensor range, all ten novice users were able to successfully navigate the robot to its goal location using a freely sketched map of the environment. Since the QPP architecture is motivated by how humans naturally communicate route directions, participants did not require extensive instructions or even knowledge of how the planning algorithm works. When allowed, users usually chose to include only a subset of the total landmarks in their sketched map, suggesting that humans neither prefer nor require representing the full environment for communicating sufficient navigation instructions. However, they were more successful at driving the robot to a specified goal location when more landmarks were included in the sketch. Despite the wide variation in sketch maps, the trajectories generated by the QPP were relatively consistent

and closely matched the desired paths.

The proposed Qualitative Path Planner offers a novel approach to semi-autonomous robot navigation and is a step in the direction towards more natural and effective human-robot interaction. Due to its flexible and intuitive nature, the QPP is a promising method for mobile robot navigation in environments for which a truth map is not available and teleoperation is not desirable. Such applications may include planetary exploration, in which large communication delays necessitate more autonomous navigation while still keeping the human operator 'in charge' of the robot, or military/rescue operations that may require teams of robots to operate in unmapped environments or areas with poor communication. Assuming a known robot sensor range, mission success can be increased by choosing appropriately spaced landmarks such that several landmarks are always within view; where identifiable landmarks are too sparse, the robot may automatically request more control by the human. This could help to limit the amount of required human interaction to times when human input is most valuable, allowing the operator to control a larger team of robots or concentrate efforts elsewhere.

CHAPTER 4

A MULTIMODAL APPROACH TO QUALITATIVE NAVIGATION USING SPEECH AND SKETCH

This chapter presents an algorithm for commanding a mobile robot trajectory using a multimodal speech and sketch interface. The user provides an approximate sketch map and verbal instructions such as “go around the chair.” Both input modes are largely unstructured, allowing novice users to provide navigation instructions naturally, as if communicating to another person. The sketch map need not be quantitatively accurate, and the speech input is not constrained to follow a rigid grammar structure. The proposed approach takes advantage of the strengths of each modality; the sketch map provides spatial relations between objects in the environment and the desired path, while verbal instructions help to identify landmarks in the map. This multimodal formulation draws on our previous work in qualitative navigation [66, 78] using both graphical inputs and verbal route instructions to provide a framework for qualitative navigation instruction that is both flexible and robust.

This chapter is organized as follows. Section 4.1 motivates the problem of communicating navigation instructions using speech and sketch, describes the dataset used for training and evaluation of the various algorithms presented, and introduces the architecture used for multimodal data integration. In Section 4.2, observations from users’ speech inputs are used to improve segmentation and recognition of sketched maps, and algorithm performance is compared to the unimodal version previously presented in [78]. An algorithm for simultaneous sketch recognition and map association is presented in Section 4.3, which is shown to further increase recognition accuracy of the sketch maps. A modi-

fied version of the Qualitative Path Planner from [66, 79] is described in Section 4.4, using both speech and sketch inputs to execute the appropriate navigation instructions. Lastly, Section 4.5 presents final discussions and conclusions.

4.1 Multimodal Strategy for Communicating Navigation Instructions

As robots become more autonomous and pervasive in the everyday lives of humans, communication between such agents and their human counterparts should be intuitive and robust, emulating the way humans interact with each other. Humans often communicate basic navigation tasks to each other using approximate spatial relationships to observable landmarks [28], without requiring a precise map (for example, “walk past the table and take a left at the elevator”). Interpreting such navigation instructions by computers has been an active area of research in recent years [20, 80, 81, 82]. On one hand, it is desirable to allow the human input to be as natural and flexible as possible. At the same time, it is necessary that efficient and reliable intention recognition be achieved. These competing objectives make it difficult to develop a robust system for qualitative robot navigation.

One proposed approach for communicating navigation instructions has been to use a graphical interface, in which the user provides an approximate map and sketches a path with respect to landmarks in the map [22, 24, 34, 35, 79]. This method allows users to clearly indicate desired spatial relations between the robot’s trajectory and objects in the environment. However, it is usually assumed that sketch recognition is trivial (e.g., landmarks are drawn as closed

polygons) and/or that the map association is known. If the sketch map is not recognized properly, or if objects in the sketch are incorrectly associated with objects in the observed environment, these approaches fail.

Natural language has also been a popular method for instructing route directions to robotic agents [80, 81, 82, 83]. In order for these approaches to be successful, however, the human's verbal instructions must generally be constrained to allowable phrases or be parsed into grammar structures such as context predicates [83] or spatial description clauses [81]. While natural language can serve as a very rich source of information for navigation instructions, truly natural speech input presents many challenges. For example, even a relatively simple statement such as, "Once you've passed it, go behind the chair on the left," could be interpreted incorrectly for several reasons:

- Pronouns are not clearly associated. (What is "it"?)
- Spatial cues are ambiguous. (Is the chair located to the left of the robot, or is the robot to pass on the chair's left?)
- Referenced objects are not unique. (What if there is more than one chair?)
- The frame of reference is unknown. (What does it mean to go "behind" something?)

To overcome some of the weaknesses associated with sketch and speech recognition, this work presents a multimodal approach for communicating navigation instructions. By combining these two communication modes into a single framework, high recognition accuracy is maintained while allowing users to interact in a truly natural way. Ambiguities in the user's speech instructions

are able to be partially or fully resolved by information contained in the sketch map, and vice versa.

Multimodal interfaces using speech and sketch have been studied and successfully implemented in a variety of applications in recent years. Simultaneous verbal and graphical data was used by [84] to improve the man-machine interface for the Airborne Warning and Control System. In [85], speech input was applied to a drawing tool, *S-tgif*, in order to minimize the number of operations, increase ease of learning, and improve operation transparency. Multimodal speech and drawing interfaces have also been used for documenting traffic accident diagrams by police [86], communicating unconstrained designs of mechanical devices [37, 87], and controlling an autonomous forklift [39].

4.1.1 System Architecture

Figure 4.1 illustrates the overall system architecture assumed in this work, where the task of understanding and executing multimodal navigation instructions has been divided into three components:

- (a) Input recognition: determine what the human spoke and/or sketched from noisy, unstructured input
- (b) Data association: assign objects in the sketch to objects in the environment observed by the robot
- (c) Navigation: the robot navigates in the true environment using the qualitative spatial relations expressed by the human

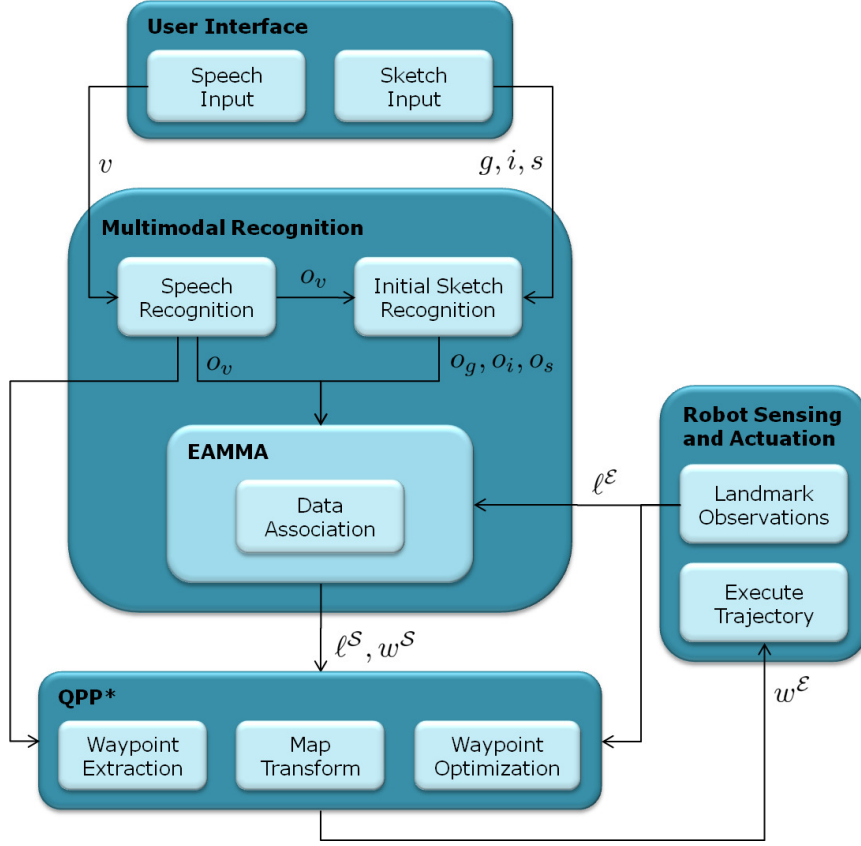


Figure 4.1: A block diagram of the system architecture. Speech and sketch inputs from a human operator are used to interpret and execute qualitative navigation instructions.

Navigation instructions are provided by the user via simultaneous sketch and speech inputs, which are collected using a microphone and computer-mouse interface. The sketch input is expressed as a sequence of gestures (g), strokes (s) and interstrokes (i), and the speech input is expressed as a sequence of verbal phrases (v). The input recognition component performs probabilistic sketch recognition to determine the *most likely* input sketch given observations extracted from pixel-level data o_g, o_s and o_i , and from recognized hypothesis phrases o_v .

The data association component requires the objects in the recognized sketched map to be associated with objects in the observed environment. For example, if the human has sketched a `chair` in the map, it must be determined to which chair in the observed environment the sketched object corresponds. This goal presents several challenges. First, a recognized object in the sketched map may not be observed in the environment, which can occur if, 1) the object does not actually exist in the true environment, 2) the object exists but has not yet been observed, or 3) the sketched gesture was recognized incorrectly, e.g., the recognized `chair` gesture is actually a `table`. Another challenge of data association is that there may be more than one of the same object in the observed environment. In general, humans tend to express route instructions by referencing relatively unambiguous landmarks [28], e.g., a human would probably not say, “take a left at the tree” if the navigator is walking through a forest. However, some amount of ambiguity must be accounted for, especially since the human instructor may not realize the instructions are vague or confusing. To address these problems, an algorithm is proposed in Section 4.3 to perform simultaneous sketch recognition and map alignment. The Evolutionary Algorithm for Multimodal Map Association, or EAMMA, aims to find the ‘best’ association between objects (also referred to as ‘landmarks’) in the observed environment $\ell^{\mathcal{E}}$ and those in the sketched map, while simultaneously improving upon the initial sketch recognition results. The output of EAMMA is the locations of sketched landmarks $\ell^{\mathcal{S}}$, some or all of which are associated with objects in the observed environment, and a series of waypoints $w^{\mathcal{S}}$ representing the sketched path.

The navigation component executes the appropriate navigation instructions using a modified Qualitative Path Planner, QPP*; the original Qualitative Path

Planner, QPP, was introduced in [66, 79]. QPP* plans a series of waypoints through the environment, $w^{\mathcal{E}}$, such that the resulting trajectory best maintains qualitative spatial relations to the landmarks in the environment $\ell^{\mathcal{E}}$, as displayed in the sketched map. QPP* expands upon the original QPP algorithm by augmenting landmark ‘importance’ along the route according to the human’s speech. For example, if the human sketches part of a path while saying “go around the chair,” a chair is likely the ‘most important’ landmark at that particular point along the trajectory. Additional details are described in Section 4.4.

4.1.2 Natural Language Processing

There are several challenges associated with natural language processing (NLP). *Voice recognition* involves processing raw audio signals into phonemes, syllables and words. Reliable relations between phonemes and their expressions in a speech signal are not always clear for several reasons. First, speech sounds are not typically strictly segmented, rather, they overlap. Second, the acoustic properties of speech sounds vary depending on the surrounding context and their position within a syllable. Third, speech rate and speaker state-of-mind (for example, excited speech versus calm speech) can significantly alter the observed speech signal. Lastly (and perhaps most obviously), the acoustic structure is significantly influenced by the specific speaker, notably the speaker’s gender, age, dialect, and accent.

Another component of NLP, often referred to as *speech recognition*, involves arranging observed phonemes into sensible words and phrases. Algorithms for speech recognition typically require both acoustic modeling and language

modeling. Hidden Markov models (HMMs) have been widely used in these applications, resulting in a high success rate for real-time speech recognition [88, 89]. Lastly, *natural language understanding* is the extraction of *meaning* from spoken or written phrases. Although verbal commands for robot control has been an active area of research in recent years, most successful applications have assumed some type of structured or pre-defined language syntax [83, 80, 81, 82].

Voice recognition and speech recognition are performed using Microsoft's Speech Software Development Kit (Speech SDK), which supports both speech recognition and speech synthesis and can be easily integrated in C#, C++, VB or any COM compliant language. The output of Speech SDK is then used in combination with the user's sketch input to perform natural language understanding in the context of qualitative mapping. Speech SDK can be used in either Command and Control or Dictation modes. In Command and Control mode, a grammar is defined containing the list of possible recognition outputs (words or sequences of words) that the recognition engine should 'listen' for and recognize. In Dictation mode, the recognition engine compares the input speech to the entire list of the dictionary words. For applications that use a limited and structured voice input, such as entering a credit card number, Command and Control mode can yield very high recognition accuracy across many users. However, Command and Control is not appropriate for unstructured natural language, as it tends to produce many false positives (i.e., spoken words not in the grammar will be wrongly recognized). While Dictation mode generally has a lower accuracy of recognition and requires that the user train the recognition engine, it does not restrict the speaker's vocabulary to a small set of words or phrases. Additionally, words and phrases recognized in Dictation mode are assigned a corresponding 'confidence' value between 0–1, which is a

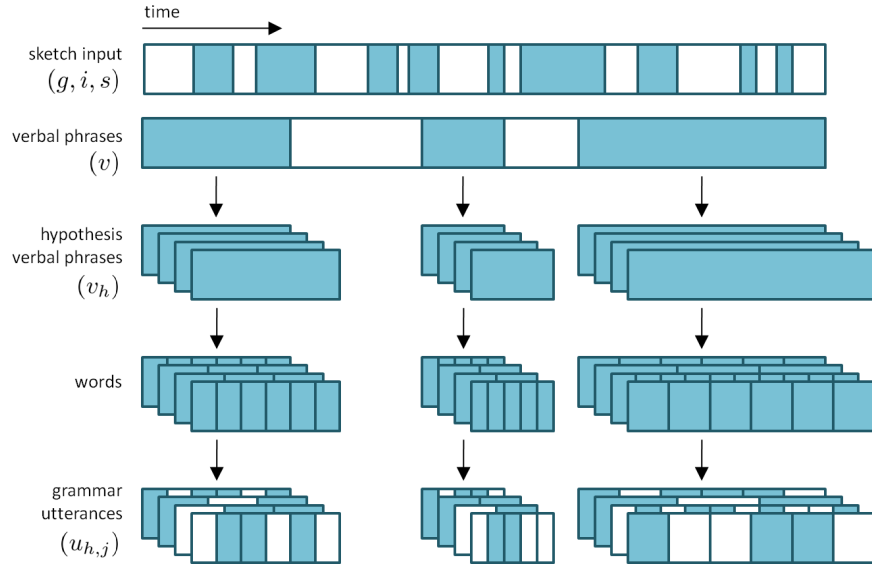


Figure 4.2: Graphical representation of natural language processing. For each verbal phrase v spoken, Speech SDK provides several hypothesis phrases v_h consisting of one or more words. Words that belong to the list of pre-defined grammar utterances $u_{h,j}$ are extracted and used for sketch recognition, map alignment, and qualitative navigation.

relative measure of the certainty of correct recognition. (It should be noted that the reported confidence score does not indicate the absolute likelihood that a word was recognized correctly, rather it provides a mechanism for comparing the relative accuracies of multiple different hypotheses for a given input.) For these reasons, Dictation mode was used for speech recognition, even though the robot will eventually only ‘listen to’ a limited vocabulary within the context of the navigation task.

Figure 4.2 shows a graphical representation of the speech processing used in this work, where shaded blocks indicate timespans during which the user is speaking or sketching. In this example, the recognition engine recognizes that the user has spoken three phrases (‘phrases’ are sequences of recognized utterances separated by a pre-defined duration of silence). For each phrase, the

Speech SDK provides several *hypothesis phrases* (up to 10) with corresponding confidence scores, which measure how well the engine’s stored pronunciation matches what the speech recognition engine heard. For example, the speech input “go past the desk and down the hall” may produce the hypothesis phrases, “go past the deck and down the wall” and “go past the desk and around them all.” Each hypothesis verbal phrase contains one or more *words*, which are also assigned confidence scores. While events raised from the recognition of verbal phrases contains timing information, specifically the phrase start time and duration of speech, individual words are not accompanied by timing information. It is therefore assumed that all words within a phrase span the same duration of time (e.g., if a phrase containing six utterances lasts three seconds, each word is assumed to have taken 0.5 seconds to speak). This is, of course, generally not true, but this assumption is necessary to extract approximate timestamps for each word, and it was not found to be severely limiting in the final results.

Lastly *grammar utterances* are extracted from the recognized words. Grammar utterances are specific words that are in the robot’s vocabulary and can be used for navigation. Specifically, the list of grammar utterances includes verbs (such as “go,” “turn,” and “stop”), objects that can be detected and used as reference landmarks (“chair,” “table,” “computer”), prepositional adverbs (“around,” “near,” “behind”), and adjectives (“tall,” “black,” “round”). To avoid constructing an overly-restrictive list of grammar utterances, a search is performed on the WordNet database [90, 91] for cognitive synonyms (synsets). For example, if “sofa” is defined as a grammar utterance, the words “couch” and “lounge” will also be added as grammar utterances.

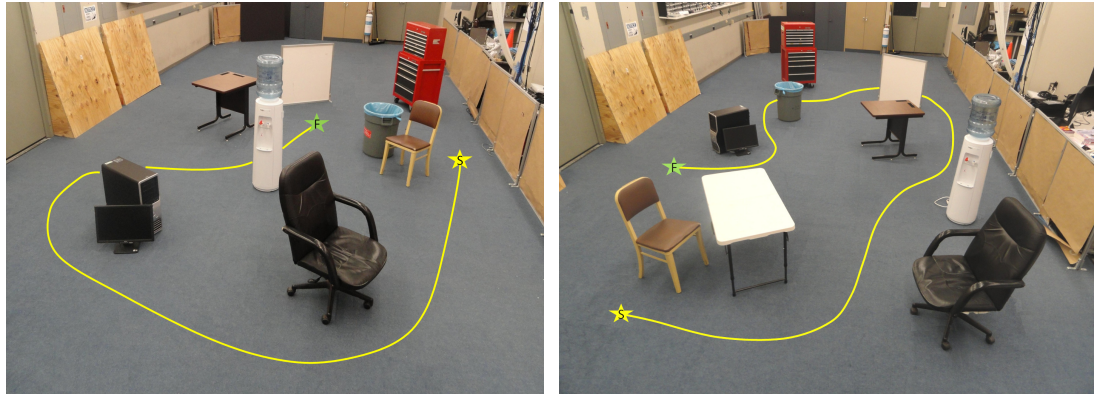
Unlike many previous approaches such as [80, 82, 83], the task of *natural*

language understanding is not directly addressed here. Rather than attempting to understand the highly unstructured and possibly confusing speech input, navigation instructions are interpreted by taking advantage of the spatial and temporal relationships between spoken utterances and sketched gestures, without the need for explicitly extracting *meaning* from the user's speech alone. This multimodal approach draws upon the strengths of each mode; speech input improves sketch recognition and data association by assigning classes to sketched objects, while sketch input disambiguates navigation instructions by visually expressing qualitative spatial relationships between the desired path and objects in the environment.

4.1.3 Dataset

To study how humans communicate navigation instructions in a multimodal fashion, a dataset was collected from 12 users. The users were Cornell University graduate students, 9 males and 3 females. Users were fluent English-speakers, however four of the twelve users spoke with noticeable foreign accents. Prior to data collection, each user spent approximately 10 minutes training the speech recognition software to improve recognition accuracy.

Users were given overhead images of six scenes consisting of various objects and a desired route indicated in yellow, as shown in Figure 4.3. Each scene included up to seven different types of objects: table, toolbox, computer, chair, water cooler, trash can, and wall. Three of the six scenes contained more than one chair and/or table. Users were told to describe the desired route by simultaneously speaking instructions into a microphone and sketching a map on a



(a) Overhead image of Scene 3.

(b) Overhead image of Scene 4.

Figure 4.3: Users were given overhead images of six scenes consisting of various objects and a desired route and asked to provide navigation instructions through the scene using speech and sketch.

Wall	Start location	Toolbox, Table, Chair, Computer	Garbage can, Water cooler

Figure 4.4: For Phases 3–4, users were instructed to sketch objects in the map as shown. Notice that the square and circle gestures are used for multiple different objects.

blank computer interface with a mouse, and were instructed to speak and sketch naturally, as if giving instructions to another person. Speech and sketch inputs for each scene were recorded as digital audio files and timestamped pixel data. Users provided these multimodal navigation instructions for each map through six phases, where each phase corresponds to a different input specification, described below:

- Phase 1 (Free Sketch and Free Speech): Users may speak and sketch freely and naturally, providing only as much information as they feel a ‘reason-

able person' would require to execute the desired trajectory.

- Phase 2 (Free Sketch and Detailed Speech): Users may sketch freely, but provide as detailed speech as possible when describing the scene (for example, instead of saying "Go around the toolbox," one might say "Go around the left of the large, red toolbox").
- Phase 3 (Semi-Structured Sketch and Free Speech): Users may speak freely, but objects in the environment must be sketched using the gestures shown in Figure 4.4.
- Phase 4 (Semi-Structured Sketch and Detailed Speech): Users must sketch objects in the environment using the gestures shown in Figure 4.4, and provide detailed speech instructions.
- Phase 5 (Very Structured Sketch and Free Speech): Users may speak freely, but objects in the environment must be sketched using the gestures shown in Figure 4.5.
- Phase 6 (Very Structured Sketch and Detailed Speech): Users must sketch objects in the environment using the gestures shown in Figure 4.5, and provide detailed speech instructions.

These six phases were chosen in order to study how speech and sketch complement and disambiguate each other. During Phases 1 and 2, users were not given specific instructions as to how to draw objects and paths in the map, discouraging uniformity across sketches and complicating the sketch recognition task. In Phases 3 and 4, several objects were to be drawn using the same gestures in order to make recognition of these objects difficult using sketch input alone. Lastly, sketch ambiguity was eliminated in Phases 5 and 6, as all objects were to be drawn using unique gestures. It was hypothesized that the incorporation of

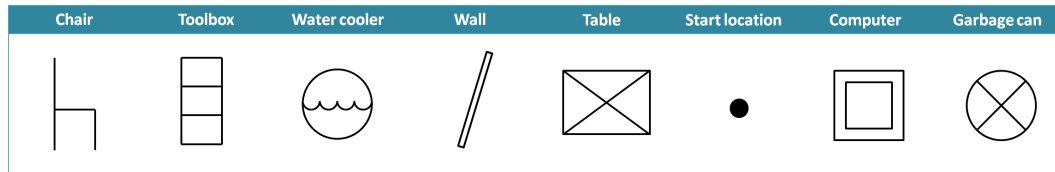
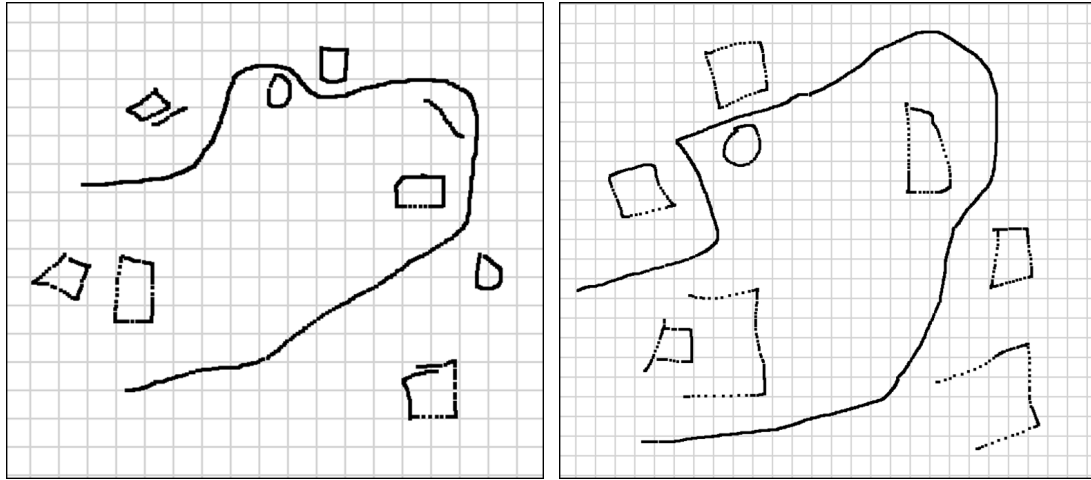


Figure 4.5: For Phases 4–5, users were instructed to sketch objects in the map as shown. Each gesture corresponds to a distinct object type.

speech input would disambiguate objects during Phases 1–4, allowing users to sketch more freely while maintaining high recognition accuracy. In Phases 1, 3 and 5, users provided “free speech,” which permitted the use of undescriptive words to describe objects and places, such as “this” and “here.” More detailed speech was used during Phases 2, 4 and 6 to help further increase recognition accuracy, as objects were to be specified by name and accompanied by adjectives.

Figure 4.6 presents two subjects’ sketched maps and corresponding speech transcriptions during Phase 1 (free sketch and free speech) for Scene 4 (the overhead image for Scene 4 is shown in Figure 4.3(b)). Several observations can be made from these inputs, and illustrate some of the challenges associated with this Phase. Both subjects first sketched the objects in the scene while verbally describing what they were drawing; they both then sketched and spoke a description of the full path through the scene. Note that not all subjects took this approach.

Both the drawing and speaking styles of these two subjects were quite different from one another. Subject 1 sketched the entire scene and represented objects as closed polygons or straight lines (and even sketched the computer as two separate objects, a CPU and a monitor), whereas Subject 6 sketched 8 of the



(a) Subject 1 speech input: "This is a chair. This is a desk. This is a CPU, this is a monitor. This is a trash can. Here's a toolbox. Here's a wall, another desk. Here's a water dispenser. Here's a chair. Start here, go around here like this, behind the trash can, in front of that, finish there."

(b) Subject 6 speech input: "Here's a chair. Here's a desk. Here's another chair. Water cooler. Wall. A dustbin, tool shack, a computer. The robot starts right here next to the chair and the desk, takes a left right next to there's another chair, passes the water cooler, goes around the desk and the wall, between and then passes through between the dustbin and the tool shack, and then between the dustbin and the computer and finishes right next to the table."

Figure 4.6: Two users' speech and sketch inputs for the same scene in Phase 1.

9 objects in the scene and drew a table and chair as open C-shaped gestures. Subject 6 gave detailed verbal instructions while drawing the path through the scene, referencing the objects in the scene as landmarks along the route. On the other hand, Subject 1 gave very vague verbal instructions such as, "Go around here like this... in front of that, finish there," relying on the sketch to communicate any spatial information required for navigation. Given the common nature of the objects in the environment, the two subjects' vocabularies were also surprisingly different; Subject 1 referred to a "CPU," "trash can," "toolbox" and "water dispenser," while Subject 6 referred to these same objects as a "com-

puter,” “dustbin,” “tool shack” and “water cooler.” Lastly, while perhaps not difficult for a human listener to understand, several phrases spoken by Subject 6 might be difficult to interpret by even sophisticated language-parsing algorithms. For example, the phrase “takes a left right next to there’s another chair” is not only grammatically incorrect, but it also contains ambiguous or seemingly conflicting instructions.

The examples in Figure 4.6 illustrate just some of the challenges encountered when developing a system to understand truly free sketched and spoken navigation instructions across multiple users. In the remainder of this work, a procedure will be described that takes advantage of the multimodal nature of a combined speech and sketch interface to overcome many of these challenges, while making few assumptions about the style or structure of users’ input.

4.2 Improving Sketch Recognition using Speech

The first component to understanding and executing multimodal navigation instructions is to determine what the human spoke and/or sketched from noisy, unstructured input (represented by the Speech Recognition and Initial Sketch Recognition blocks in Figure 4.1). As opposed to previous methods, the approach presented here does not constrain users to speaking pre-defined allowable phrases [39], nor does it attempt to map the speech input into rigid grammatical structures for extracting useful meanings [43, 44, 46]. Rather, this work expands upon [78] to improve sketch recognition by incorporating low-level observations from the user’s speech input. Instead of interpreting the user’s full speech input in the sense of *intention recognition*, individual words in the user’s

speech input are used to increase stroke and gesture classification accuracy. The sketch recognition problem is framed as a variable duration hidden Markov model, which supports flexible and multi-stroke gestures. The model is learned from training data, largely avoiding the need to explicitly define an expected structure of either sketch or speech input. This probabilistic framework helps shift the burden of recognition from the user to the machine, allowing the user to speak and sketch more naturally.

4.2.1 Probabilistic Sketch Representation

Segmentation is the process of determining which stroke(s) belong to which gesture, and *recognition* is the assignment of gestures into meaningful classes. In this work, the ambiguous transition multi-stroke gesture problem is considered, where segmentation and recognition must be performed simultaneously. A sketch is characterized as a sequence of strokes, gestures, and interstrokes, defined as follows:

- **Stroke:** A stroke s_j is a collection of pixels generated during a single pen-down (or mouse-down) instance. Each stroke corresponds to a particular stroke class $s_j = c_s \in \bar{S}$, where the set of M_s possible stroke classes \bar{S} is finite and known.
- **Gesture:** A gesture g_j is a complete shape or symbol composed of d_j strokes, where $d_j \in \mathbb{N}_1$ is the (unobserved) gesture duration. Each gesture corresponds to a particular gesture class $g_j = c_g \in \bar{G}$, where the set of M_g possible gesture classes \bar{G} is finite and known.
- **Interstroke:** An interstroke i_j is the transition between two consecutive

strokes. The corresponding interstroke class $i_j = c_i \in \bar{I}$ defines whether a stroke belongs to the same gesture as the previous stroke (i.e., a self-transition on the gesture) or is the start of a new gesture.

- **Sketch:** A sketch is a series of N_S strokes and N_G gestures $G = [g_1, \dots, g_{N_G}]$, where the total number of gestures drawn N_G is unknown (because d_j is unobserved).

As in [78], each stroke is assumed to belong to only one gesture and that strokes are not interspersed, i.e., users do not return to a gesture after starting a new one. The total number of strokes in a complete sketch $N_S = \sum_{j=1}^{N_G} d_j$ is assumed to be known, and stroke start/end points are identified by mouse down/up instances. The stroke classes in \bar{S} coincide with the gesture classes in \bar{G} (therefore $M_s = M_g = M$); for example, if g_j is a `chair` gesture, the strokes composing that gesture S_j are `chair` strokes.

This work expands upon [78] by introducing speech as a second input mode from the human. As the user speaks, the verbal phrase v is recognized as one or more hypothesis phrases v_h , the observations of which, o_{v_h} , are functions of the recognized grammar utterances $u_{h,j}$ and the associated belief (confidence value) b_{v_h} .

Figure 4.7 illustrates a graphical representation of an example speech and sketch input, where shaded nodes represent observations, unshaded nodes represent hidden (unobserved) variables, and arrows represent conditional dependencies. In this example, the first gesture g_1 has a duration of $d_1 = 3$ strokes, and the second gesture g_2 has a duration of $d_2 = 2$ strokes. Interstroke nodes i_1 , i_2 , and i_4 represent self-transitions on the gestures, while node i_3 represents a transition from gesture g_1 to gesture g_2 . Speech recognition of the verbal input

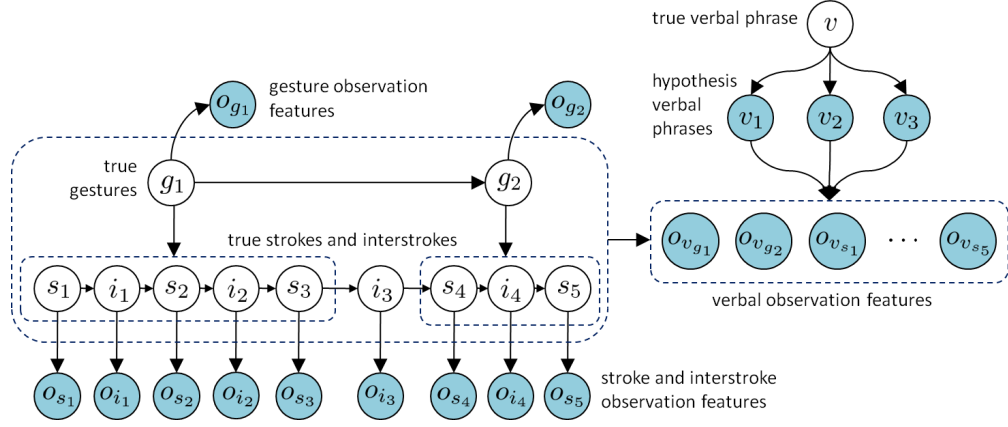


Figure 4.7: Graphical representation of an example speech and sketch input, where the sketch consists of five strokes $s_j, j \in [1, \dots, 5]$ and two gestures g_j . The user's speech v produces three hypothesis phrases v_j

v produces three hypothesis phrases $v_{1:3}$.

For each gesture, a set of gesture observation features o_g are extracted from pixel-level data and verbal observation features o_{v_g} are extracted from the hypothesis phrases. Similarly for each stroke, a set of gesture observation features o_s are extracted from pixel-level data and verbal observation features o_{v_s} are extracted from the hypothesis phrases. Lastly, interstroke observation features o_i are extracted from pixel-level data for each interstroke. The goal is to correctly segment and recognize the sketched map, given all observations extracted from the speech and sketch inputs. This is accomplished by maximizing the observation likelihood:

$$p(\text{sketch}|O) = p(G, S, I|O) \quad (4.1)$$

Here, $O = \{O_G, O_S, O_I, O_V\}$ are the observations on the gestures, strokes, interstrokes, and verbal phrases respectively. Equation 4.1 can be factorized as:

$$p(G, S, I|O) = p(S, I|O_S, O_I, O_{V_S}) p(G|S, I, O_G, O_{V_G}) \quad (4.2)$$

Assuming a first-order Markov model for strokes and interstrokes, and uniform marginal distributions for all gesture, stroke, and interstroke classes, the terms on the right-hand side of Equation 4.2 can be written as:

$$p(S, I | O_S, O_I, O_V) \propto p(s_1 | o_{s_1}, o_{v_{s_1}}) \prod_{j=1}^{N_S-1} p(s_{j+1} | i_j, s_j) p(s_{j+1} | o_{s_j}, o_{v_{s_{j+1}}}) p(i_j | o_{i_j}) \quad (4.3)$$

$$\begin{aligned} p(G | S, I, O_G, O_{V_G}) &= \prod_{j=1}^{N_G} p(g_j | S_j, I_j, o_{g_j}, o_{v_{g_j}}, g_{j-1}) \\ &\propto \prod_{j=1}^{N_G} p(g_j | S_j, I_j) p(g_j | o_{g_j}, o_{v_{g_j}}) p(g_j | g_{j-1}) \end{aligned} \quad (4.4)$$

where S_j and I_j are the strokes and interstrokes that compose gesture g_j .

In this work, a few assumptions are made before training the model to reduce the number of distributions in Equations 4.3 and 4.4 that must be learned and the amount of training data required. First, the probability distribution of transitioning from one gesture class to another, $p(g_j | g_{j-1})$, is assumed to be uniform. Second, gesture ‘mistakes’ or otherwise unknown gestures are not considered, although this could also be easily incorporated into the framework by including an $(M + 1)$ th other gesture class. Third, gesture g_j must be of the same class as its associated strokes S_j (e.g., a chair can only be drawn with chair strokes). Fourth, if i_j is of the class same gesture (as opposed to new gesture) then $s_{j+1} = s_j$, otherwise $p(s_{j+1} | i_j, s_j)$ is uniform. Lastly, while the number of gestures N_G in the sketched map is unknown, the total number of strokes N_S is known (i.e., the beginning and end of the sketch is unambiguous).

The remaining terms in Equations 4.3 and 4.4 must still be defined: $p(s_1 | o_{s_1}, o_{v_{s_1}})$, $p(i_j | o_{i_j})$, and $p(g_j | o_{g_j}, o_{v_{g_j}})$. As in [78], each of these conditional probabilities is treated as a multinomial classification problem, the distributions of which are determined using a supervised learning algorithm, Sparse Multi-

nomial Logistic Regression (SMLR) [55]. SMLR is a true multiclass formulation based on multinomial logistic regression [56], i.e., it doesn't reduce the full multiclass problem into multiple binary classification problems. Instead, it learns weight vectors \mathbf{w} such that the likelihood of y being classified as class c for a set of observed features \mathbf{x} is given by:

$$\mathcal{L}_c(y) \equiv p(y = c | \mathbf{x}; \mathbf{w}) = \frac{\exp(\mathbf{w}_c^T \mathbf{x})}{\sum_{j=1}^m \exp(\mathbf{w}_j^T \mathbf{x})} \quad (4.5)$$

where m is the total number of possible classes. In this work, y corresponds to g_j , s_j , or i_j and \mathbf{x} corresponds to the observation features, described in the following sub-sections.

Gesture, Stroke, and Interstroke Observation Features

The observation features o_g , o_i and o_s are a subset of those used in [78]. For each full sketched gesture, 53 features (corresponding to o_{g_j}) were extracted from the pixel data (features 0–52 in Table A.2 of the Appendix). Some of these features were adopted from a portion of the g-48 set presented in [59] and [60], which was developed to be generally well-suited for identifying multiple-stroke gestures. Also included in o_{g_j} are features corresponding to initial orientation angles, the amount of time spent drawing each gesture, the total number of strokes, and clockwise/counter-clockwise orientation. An additional set of features, referred to as Histogram of Tangents (HoT features), and were presented in [78]. These features are defined by calculating the stroke tangent angle between each pair of neighboring pixels. The angles are discretized into eight bins, normalized to 1, and 'shifted' such that the first bin is aligned with either the dominant angle (defining eight global HoT features) or the initial angle

(defining another eight global HoT features).

A total of 50 features were extracted as stroke observations o_{s_j} , and were the same as those used for gestures (minus three redundant features corresponding to number of strokes, average time drawing each stroke, and total time drawing the gesture). Nine interstroke features (o_{i_j}) were defined by extracting information from consecutive strokes, including relative positions, sizes, and orientations, as well as temporal information, and are listed in Table A.1 of the Appendix.

Verbal Observation Features

Several previous approaches to multimodal recognition of speech and sketch only consider single-speech and single-gesture inputs, where associations between spoken words and sketched gestures are unambiguous. For example, [37] defines a temporal window that extends a time $\Delta T = 3\text{sec}$ before and after a stroke, and assumes that any words that fall at least partially within that window are associated with the stroke. By contrast, this work expands upon [78] by introducing a set of verbal observation features o_v , which are learned and used for probabilistic inference along with o_g , o_i and o_s . This set of features encapsulates spatial and temporal relationships between spoken words or word groups and sketched gestures.

Spoken words, or more specifically *grammar utterances*, are assigned to one or more *word groups* W that captures some common functional characteristic. For example, a word group may represent parts of speech (nouns, verbs, etc.), may define some physical trait (colors, sounds), or may correspond to a particu-

lar sysnet (i.e., “sofa,” “couch” and “lounge”). The 14 word groups used in this work are five parts of speech and nine gesture classes $c_g \in \bar{G}$: table, toolbox, computer, chair, water cooler, trash can, wall, path, or start location. For each stroke (or gesture), the following observation features o_v are extracted:

- t_{\min} : minimum time to each word group
- d_{\min} : minimum distance to each word group
- p_{tw} : time-weighted probability for each word group
- p_{dw} : distance-weighted probability for each word group

The minimum time (t_{\min}) and minimum distance (d_{\min}) to a word group W is defined as:

$$t_{\min}(W) = \min_{u \in W} [\Delta t_u] \quad (4.6)$$

$$d_{\min}(W) = \min_{u \in W} [\Delta d_u] \quad (4.7)$$

where Δt_u is the minimum difference in time between when the stroke (or gesture) was sketched and the utterance u was spoken, and Δd_u is the minimum distance in pixels between the stroke (or gesture) and the pixel drawn at the time utterance u was spoken. These features measure cross-modality coherence; people operating in multiple modes tend to synchronize their inputs, i.e., they do not talk about one topic while sketching another [87].

The time-weighted probability (p_{tw}) and distance-weighted probability (p_{dw}) of a word group W are defined as:

$$p_{tw}(W) = \sum_h \sum_{u_{h,j} \in W} b_{u_{h,j}} b_{v_h} \left(1 - \frac{\Delta t_{u_{h,j}}}{T}\right)^3 \quad (4.8)$$

$$p_{dw}(W) = \sum_h \sum_{u_{h,j} \in W} b_{u_{h,j}} b_{v_h} \left(1 - \frac{\Delta d_{u_{h,j}}}{D}\right)^3 \quad (4.9)$$

where v_h is the h^{th} recognized hypothesis phrase, $u_{h,j}$ is the j^{th} utterance (i.e., word) in the h^{th} phrase, b_{v_h} is the reported belief (or confidence) for the phrase v_h , and $b_{u_{h,j}}$ is the reported belief for utterance $u_{h,j}$. T is the timespan of the full speech input, and D is the length span of the total sketch (i.e., the distance between the two farthest sketched pixels). The cubic terms in these features are also measures of cross-modality coherence, but they are weighted by the corresponding utterance and phrase confidence values. This tends to make $p_{tw}(W)$ and $p_{dw}(W)$ large for the word groups containing recognized utterances spoken close (in time or space) to an object sketched in the map, but only if those utterances are recognized with high confidence.

Features 4.6–4.9 described above encapsulate spatial and temporal relationships between sketched strokes and spoken words, without the need to set arbitrary thresholds for associating inputs from each modality. The spatial features $d_{\min}(W)$ and $d_{wp}(W)$ are particularly useful for interpreting navigation maps, as users frequently sketch objects in the environment first, and make verbal reference to the objects later as the path is being drawn (for example, “go around this *chair* here.”) Temporal features alone would likely not enable these late utterances to be properly associated with the intended corresponding objects. The list of verbal observation features used in this work can be found in Table A.3 in the Appendix.

4.2.2 Multimodal Recognition Results

Three gesture classifiers were learned using SMLR: $p(g_j = c_g | o_{g_j})$, $p(g_j = c_g | o_{v_{g_j}})$, and $p(g_j = c_g | o_{g_j}, o_{v_{g_j}})$, where $c_g \in \bar{G}$ can take one of nine classes: table, toolbox, computer, chair, water cooler, trash can, wall, path, or start location. A 5-fold cross-validation was performed using 50 examples of each stroke class, taken from the dataset. The average gesture classification accuracy for each phase of data collection is plotted in Figure 4.8(a). Similarly, three stroke classifiers were learned: $p(s_j = c_s | o_{s_j})$, $p(s_j = c_s | o_{v_{s_j}})$, and $p(s_j = c_s | o_{s_j}, o_{v_{s_j}})$, where $c_s \in \bar{S}$ can take the same nine classes. 5-fold cross-validation was performed using 50 examples of each stroke class, taken from the dataset. The average stroke classification accuracy for each phase of data collection is plotted in Figure 4.8(b). One interstroke classifier was learned using only sketch observations, as in [78]: $p(i_j = c_i | o_{i_j})$. This classifier performed approximately the same across all phases with about 90% accuracy. Note that for all classifiers, the same multimodal data was used; for $p(g_j = c_g | o_{v_{g_j}})$ and $p(s_j = c_s | o_{v_{s_j}})$ the sketch input was ‘ignored’, and for $p(g_j = c_g | o_{g_j})$ and $p(s_j = c_s | o_{s_j})$ the speech input was ‘ignored.’ The navigation instructions provided would likely be different if the users operated in truly sketch-only and speech-only modes.

Several interesting results are plotted in Figure 4.8. First, both gesture and stroke recognition accuracy was around 70–80% using verbal observation features alone. These results are not surprising, as verbal observation features o_{v_g} and o_{v_s} encode spatial and temporal relationships between sketched gestures/strokes and spoken utterances; since users did not always verbally describe *what* they were sketching *when* they sketched it (in fact, some users failed

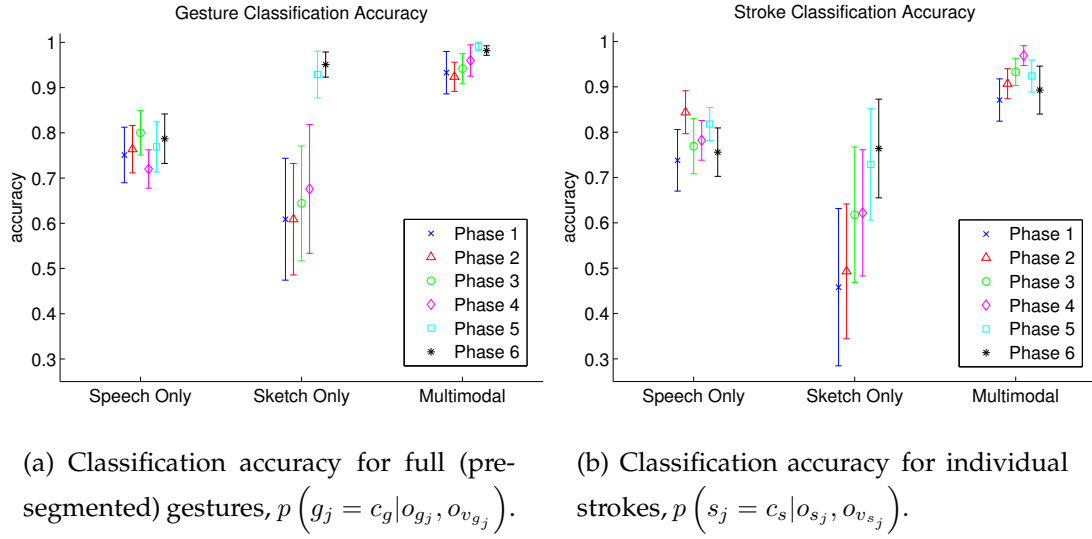


Figure 4.8: Classification accuracy for both gestures and strokes significantly improve by using multimodal observation features rather than speech or sketch alone.

to verbally describe what they were sketching at all), these features are not always indicative of the gesture or stroke class. Second, recognition accuracy using only verbal observation features was not significantly different across the six phases. Although users were told to give very detailed verbal instructions during Phases 2, 4 and 6, the biggest difference was the increased use of adjectives; for example, the toolbox was often referred to as a “large red toolbox.” Since the utterances “large” and “red” are not specifically associated with class `toolbox` (rather, they are members of the Adjective word group), these additional descriptions did not prove to be very useful for classifying sketches. This could, however, be incorporated into the framework by allowing adjectives to be probabilistically associated with specific objects.

Using only sketch features o_g , gesture classification accuracy was lowest for Phases 1–4, between 60–70%. This is expected, as users were not instructed how

	ta	to	co	ch	wc	tr	wa	pa	st		ta	to	co	ch	wc	tr	wa	pa	st
table	0.44	0.04	0.08	0.36	0	0	0.08	0	0	table	0.88	0.04	0.04	0	0	0	0.04	0	0
toolbox	0.08	0.52	0.2	0.08	0.04	0.04	0.04	0	0	toolbox	0	0.96	0.04	0	0	0	0	0	0
computer	0.2	0.04	0.44	0.24	0.04	0	0.04	0	0	computer	0	0	0.96	0.04	0	0	0	0	0
chair	0.08	0.28	0.2	0.28	0	0.12	0.04	0	0	chair	0.04	0	0	0.96	0	0	0	0	0
water cooler	0	0	0	0	0.6	0.4	0	0	0	water cooler	0	0	0	0	1	0	0	0	0
trashcan	0	0	0	0	0.28	0.72	0	0	0	trashcan	0	0	0	0	0	1	0	0	0
wall	0.08	0	0	0	0	0.04	0.8	0.08	0	wall	0.04	0	0	0	0.04	0	0.92	0	0
path	0	0	0	0	0	0	0	1	0	path	0	0	0	0	0.12	0	0.04	0.8	0.04
start	0	0	0	0	0	0	0	0	1	start	0	0	0	0	0	0	0	0	1

(a) Confusion matrix for gestures in Phase 3 using only sketch features.

(b) Confusion matrix for gestures in Phase 3 using sketch and speech features.

Figure 4.9: Gesture classification accuracy in Phase 3 using sketch features alone, due to the ambiguous nature of the gestures. Four object classes were sketched as squares, and two classes were sketched as circles. Speech observation features help to disambiguate these gestures, significantly increasing classification accuracy.

to draw gestures during Phases 1–2, so there was not necessarily any continuity across users’ sketches, nor distinction between gesture classes. Similarly, during Phases 3–4, several gestures were intentionally ambiguous (e.g., *water cooler* and *trash can* were both drawn as circles). The confusion matrix in Figure 4.9(a) confirms that most errors in gesture classification occurred between gestures that were represented by the same symbol. For Phases 5–6, where each gesture class was represented by a distinct symbol, gesture recognition accuracy was much higher, around 94%. The results of a Wilcoxon rank sum test show that the difference in the number of correctly recognized gestures between Phases 1–4 and Phases 5–6 was statistically significant ($p < 0.001$).

The results of stroke classification shown in Figure 4.8(b) followed a similar

trend, although recognition accuracy was consistently lower than that of gesture classification. Even during Phases 5–6, stroke classification accuracy is approximately 75%, because there is less information encoded in single strokes, many of which are ambiguous straight lines or circles.

The most significant result is that both gesture and stroke classification accuracies are greatly improved in the multimodal case for all phases. By incorporating both speech and sketch observations, gestures are recognized on average with an accuracy of 92–98% and strokes with an accuracy of 87–97%. The results of a Wilcoxon rank sum test show that gesture and stroke recognition accuracies are statistically significantly better in the multimodal case than both the speech-only and sketch-only cases (all $p < 0.001$).

Figure 4.9(b) plots the confusion matrix for Phase 3 using both speech and sketch inputs, and confirms the hypothesis that natural speech can help resolve ambiguities in the sketched maps. Furthermore, these results reinforce the idea that speech can be successfully incorporated into a multimodal framework without the need to explicitly extract meaning from the speech input or to define a temporal window during which words and strokes should be associated.

Multi-Stroke Sketch Recognition

The results shown in Figure 4.8(a) are for individual pre-segmented gestures, i.e., the sketch has already been separated into separate complete gestures and there is no ambiguity regarding which strokes belong to which gestures. However, the probabilistic approach proposed here does not assume such unam-

biguous gesture transitions. Therefore, in order to recognize a full multi-stroke sketch map, one must perform simultaneous segmentation and classification. Using the method presented in [78], a forward tree-search algorithm is implemented to find the most likely sketch by searching for the sequence of strokes, interstrokes and gestures that maximizes Equation 4.2. (Refer to [78] for algorithm details.)

With respect to sketch recognition, four performance metrics are considered. First, *stroke recognition accuracy* represents the percent of strokes in a sketch that are correctly grouped into appropriate gestures and classified as the correct class. Second, *sketch exactness* is the percent of full sketches perfectly recognized without any errors. *Algorithm optimality* indicates whether the sketch recognition algorithm is optimal, in the sense that it is guaranteed to return the result that maximizes the observation likelihood. Lastly, *algorithm efficiency* measures the amount of computation resources consumed, namely with respect to speed and scalability.

The full sketch recognition algorithm presented in [78] was performed on 51 sketched maps. The size of the tree search grows exponentially with the number of strokes, therefore only maps drawn with 10 strokes were analyzed. Although optimal, this recognition strategy becomes impractical as the number of strokes in the sketch increases, however it is presented here for completeness and is briefly compared to a newly proposed algorithm in Section 4.3.

For the 51 maps analyzed, the segmentation and recognition algorithm presented in [78] yields an average stroke recognition accuracy of 87%, meaning 87% of all strokes in the sketch are correctly grouped into appropriate gestures and classified as the correct class. However with respect to sketch exact-

ness, only 24 of the 51 sketches were perfectly recognized without any errors. Since the sketches represent maps that will eventually be used for navigation, any errors in sketch recognition may have devastating consequences. Errors in segmentation (e.g., two `chair` gestures are incorrectly grouped into a single `chair`) or classification (e.g., a `chair` gesture is incorrectly recognized as a `table`) may prevent the robot from successfully navigating the desired route. In an effort to further reduce these errors, Section 4.3 presents an algorithm to improve sketch recognition by associating objects in the observed environment with those indicated in the sketched map.

Consider the following scenario. One person is explaining to another person how to get to a particular location in town. The guide draws an approximate map while speaking instructions such as, “There’s a bank here, take a right at the Mexican restaurant, and go past the park...” When finished receiving the navigation instructions, the navigator sets off and follows the indicated route. Along the way, he notices there to be an error in the map – he observes an Indian restaurant where the guide has indicated a Mexican restaurant is located. Assuming everything else in the map matches the observed environment and the navigator does not seem to be lost, he may conclude that either a) the speaker made a mistake, or b) he misunderstood the speaker. By taking advantage of observations of the landmarks referenced in the navigation instructions, the navigator may be able to augment his interpretation of the map and re-label the restaurant.

This type of situation is often encountered in human communication, as humans are prone to making mistakes both in giving and receiving instructions. It is reasonable to conclude that map recognition can be improved by incor-

porating observations of the surrounding environment. This motivates a new approach proposed in Section 4.3, which performs simultaneous sketch recognition and landmark association.

4.3 Evolutionary Approach to Simultaneous Sketch Recognition and Map Alignment

The second component in Figure 4.1 to understanding and executing multimodal navigation instructions is to associate objects in the sketch map to objects with the observed environment. For example, if the human sketches a path around an object while saying “go around the chair,” the robot must be able to determine which object in its environment is the referenced “chair” (perhaps there is more than one), and execute the desired trajectory with respect to that chair according to the drawn path. A similar problem was addressed by [20], where the object correspondence between a sketched map and occupancy grid map (OGM) is accomplished using an Evolutionary Algorithm for Scene Matching (EASM). Spatial relations between different objects in a scene are captured using the histogram of forces method, and an evolutionary algorithm is used to find the best histogram relational map. Objects in the sketched map and may differ in terms of orientation, translation, shape and size.

The approach presented here differs from [20] in several ways. First, the sketched maps used by [20] are pre-segmented, i.e., sketched objects are defined as closed polygons and are extracted prior to map-matching. By contrast, the approach proposed here does not assume pre-segmented sketches; it is not even known which parts of the sketch correspond to objects as opposed to paths,

and instead aims to solve the segmentation, recognition, and map alignment problems simultaneously. Also, the approach in [20] assumes one-to-one object mapping, i.e., each object in the OGM is represented by *at most* one polygon in the sketch. Here, since objects in the sketch are not pre-segmented, each object in the observed environment may be associated with any number of strokes, or possibly none. Lastly, whereas [20] performs map-matching using only spatial relations between objects, the algorithm presented here also incorporates object classification in order to improve object association.

4.3.1 EAMMA: An Evolutionary Algorithm for Multimodal Map Association

The task of simultaneous sketch recognition and map alignment requires optimization of three different (and often competing) objectives. First, strokes and gestures in the sketch should be classified with high likelihood (i.e., a gesture that *looks* like a car should probably be classified as a `car`). Second, objects in the sketched map should be assigned such that their relative spatial relationships are similar to those in the observed environment (e.g., if object A is located between objects B and C in the observed environment, the gesture representing object A in the sketch should also be drawn between objects B and C). Lastly, the number of associations should be maximized (i.e., as many objects/stroke matchings should be made as possible). Here, a novel algorithm is proposed to solve for ‘good’ associations between strokes in a sketched navigation map and objects in the observed environment. The Evolutionary Algorithm for Multimodal Map Association, or EAMMA, performs map-matching by assigning

a score, or fitness, to a recognized sketch based on how well it achieves these objectives, and searching for the solution with the highest fitness.

Evolutionary algorithms (EA) perform search and optimization by generating a population of solutions and evolving the population using fitness-based selection and repetitive reproduction operations. They have been applied in a wide variety of domains from task scheduling [92], to molecular geometry optimization [93], to self-modeling robots [94]. EAs are particularly useful for types of problems that are fully or partially separable, those which are NP-hard, and those that have very large search spaces. They are not guaranteed to converge to the optimal solution in finite time, but can often find the optimal (or a ‘good enough’) solution faster than brute-force methods [95]. EAs are also appropriate for performing multiobjective optimization, as they give rise to a set of optimal solutions, known as Pareto-optimal solutions, rather than a single optimal solution [96].

For the variation of EAMMA presented here, several assumptions are made. First, it is assumed that the robot is able to detect object classes, e.g., it knows whether it is observing a `chair` or a `computer`. In practice, this may be accomplished using various estimation techniques for recognizing object classes [97, 98, 99]. Second, it is assumed that the robot has observed most (if not all) of the environment, i.e., all the objects in the sketched map have been located. Third, it is assumed that the sketched map contains most (if not all) of the detectable objects in the environment, i.e., the user has not omitted objects from the map that the robot can observe. Lastly, once observed, the locations of all objects are assumed to be known with respect to some fixed reference frame. Some of these assumptions could be relaxed by modifying the fitness function

used by EAMMA, for example by incorporating uncertainties in object classes and/or locations, but these are not considered here.

Genetic Representation

The goal of EAMMA is to determine which sketched strokes correspond to which objects in the observed environment. Each individual in the population represents a potential map association solution and is encoded as a string of $3N_S$ values. Figure 4.10 illustrates an example individual. In this case, there are $N_S = 13$ strokes in the sketched map. The associated object string (top row in Figure 4.10) encodes the object association assignments for each stroke. In this case, the first three strokes are associated with object 5, the next two strokes are assigned to object -1 (an assignment of -1 indicates ‘no object’, i.e. the stroke represents a `path` or an otherwise unobserved object), the next two strokes are associated with object 4, and so on. The stroke label string (second row in Figure 4.10) encodes the label or class of each stroke. For most strokes, this is simply the class of the associated object; e.g., if a stroke is associated with a `chair` object, the stroke class is `chair`. Every stroke must have a stroke class, even if it does not have an associated object (for example, a stroke may be assigned to an unobserved `table`). The third row in Figure 4.10 encodes whether the stroke is the start of a new gesture (value = `true`) or is the continuation of the previous gesture (value = `false`). For the example shown in Figure 4.10, bold red lines indicate transitions between successive gestures. This genetic representation fully describes a map association solution, allowing for both unassociated objects (e.g., in Figure 4.10 no stroke is associated with object 1) and unassociated strokes (strokes 4 and 5 are not associated with any object in the environment).

length = # of strokes

associated object	5	5	5	-1	-1	4	4	2	-1	6	6	6	3
stroke label	2	2	2	7	7	1	1	5	9	7	7	7	4
new gesture	T	F	F	T	F	T	F	T	T	T	F	F	T

Figure 4.10: The genetic representation of an example individual, which encodes associations between strokes in the sketched map and objects in the observed environment.

Fitness Function

Individuals in the current population are selected with probability proportional to their fitness, which is evaluated as a combination of four functions:

$$F = f_c f_p f_{ma} f_a \quad (4.10)$$

where f_p is the path existence fitness, f_c is the classification fitness, f_{ma} is the map alignment fitness, and f_a is the association fitness. The path existence fitness $f_p = 1$ if at least one stroke is classified as a path, and 0 otherwise. This removes individuals from the population that do not include a path as part of the navigation map. The classification fitness f_c is the average likelihood of all stroke, gesture, and interstroke classifications:

$$f_c = \left(\frac{1}{N_S} \sum_{j=1}^{N_S} p(s_j = c_{s_j}) \right) \left(\frac{1}{N_G} \sum_{j=1}^{N_G} p(g_j = c_{g_j}) \right) \left(\frac{1}{N_S - 1} \sum_{j=1}^{N_S - 1} p(i_j = c_{i_j}) \right) \quad (4.11)$$

This encourages sketched strokes to be associated with objects whose corresponding classes are more likely. For example, if stroke s_2 has a high likelihood of belonging to class `chair`, then the $p(s_j = c_{s_j})$ term of f_c will be larger for any individual that has assigned s_2 to a `chair` object in the environment.

The map alignment fitness f_{ma} is evaluated by calculating the transformation matrix $T^{\mathcal{E} \rightarrow \mathcal{S}}$ using ordinary least squares:

$$T^{\mathcal{E} \rightarrow \mathcal{S}} = \left((X^{\mathcal{E}})^T X^{\mathcal{E}} \right)^{-1} (X^{\mathcal{E}})^T X^{\mathcal{S}} \quad (4.12)$$

where:

$$X^{\mathcal{E}} = \begin{bmatrix} \bar{\ell}_{1,x}^{\mathcal{E}} & \bar{\ell}_{1,y}^{\mathcal{E}} & 1 \\ \vdots & \vdots & \vdots \\ \bar{\ell}_{m,x}^{\mathcal{E}} & \bar{\ell}_{m,y}^{\mathcal{E}} & 1 \end{bmatrix} \quad (4.13)$$

$$X^{\mathcal{S}} = \begin{bmatrix} \ell_{1,x}^{\mathcal{S}} & \ell_{1,y}^{\mathcal{S}} & 1 \\ \vdots & \vdots & \vdots \\ \ell_{m,x}^{\mathcal{S}} & \ell_{m,y}^{\mathcal{S}} & 1 \end{bmatrix} \quad (4.14)$$

where each sketched landmark $\ell_j^{\mathcal{S}}$ represented by a single gesture is associated with observed object $\ell_j^{\mathcal{E}}$. The map alignment fitness f_{ma} is the inverse average distance between each observed landmark $\ell_j^{\mathcal{E}}$ and the associated sketched landmark transformed into environment coordinates $\ell_j^{\mathcal{S}} (T^{\mathcal{E} \rightarrow \mathcal{S}})^{-1}$:

$$f_{ma} = \left(\frac{1}{m} \sum_{j=1}^m \left\| \bar{\ell}_j^{\mathcal{E}} - \ell_j^{\mathcal{S}} (T^{\mathcal{E} \rightarrow \mathcal{S}})^{-1} \right\|_2 \right)^{-1} \quad (4.15)$$

where m is the number of associated landmarks/strokes. This function encourages gestures to be assigned such that the sketched objects have a similar spatial arrangement to that of the observed objects in the environment. To prevent this term from becoming too large and dominating the fitness function F , f_{ma} is limited to being ≤ 1 . This effectively penalizes for sketched landmarks that, after transforming into environment coordinates, are located more than 1m from the observed landmark. Without setting this limit, f_{ma} would tend to drive all but 3 objects to remain unassigned, allowing $\bar{\ell}_j^{\mathcal{E}} = \ell_j^{\mathcal{S}} (T^{\mathcal{E} \rightarrow \mathcal{S}})^{-1}$ for all $j = 1, 2, 3$ and therefore $f_{ma} = \infty$.

The association fitness f_a assigns a penalty for each stroke that is unassigned to an object in the environment and each object that is unassigned to a stroke:

$$f_a = 0.5^{(N_{us} + N_{uo})} \quad (4.16)$$

where N_{uo} is the total number of unassigned objects and N_{us} is the total number of unassigned strokes (except for `path` and `start` strokes, which are not associated with physical objects in the environment). This has the effect of promoting individuals that represent fully-assigned sketches (every object in the environment exists in the sketch, and vice-versa). This function may be adjusted to account for partially-observed environments (e.g., the robot has limited sensor range and has not explored the entire environment) or if the robot's environment representation is expected to be much richer than the sketch (e.g., the robot is able to observe and map many landmarks that are not needed for the navigation task). However, in this work it is assumed that a sketched map includes most (if not all) objects in the environment.

Initialization

The choice of initial population can greatly influence the accuracy and/or speed of a genetic algorithm. Traditionally, the initial population is generated randomly, allowing the entire search space to be uniformly covered. However, 'seeding' the population with individuals in areas where optimal solutions are likely can improve the algorithm's efficiency. Here, half the initial population is generated randomly by first assigning each stroke to be the start of a new gesture with probability 0.5, otherwise it is assigned to be the continuation of the previous gesture. This segments the sketch into complete gestures and defines

the third row in Figure 4.10. Next, each gesture is randomly associated with an object in the environment or to ‘no object’ with uniform probability, encoded in the first row in Figure 4.10. If the gesture is associated with an object, it assumes the object’s class; if the gesture is associated with ‘no object’ its class is randomly chosen. These object classes are encoded in the second row in Figure 4.10.

The other half of the initial population is generated by first segmenting the sketch by defining each interstroke i_j as a `gesture transition` with probability $p(i_j = GT | o_{i_j})$. This defines which strokes are the start of a new gesture, or the continuation of a previous gesture (third row in Figure 4.10). Next, each segmented gesture g_j is assigned to object class c_g with probability $p(g_j = c_g | o_{g_j}, o_{v_{g_j}})$ (second row in Figure 4.10). Lastly, each gesture (chosen at random) is associated to objects in the environment. If an unassociated object in the observed environment exists with the corresponding object class (e.g., g_j has been assigned the class `chair`, and there is a chair in the environment which has not already been associated to a sketched gesture), g_j is associated with that object. Otherwise, g_j is assigned to ‘no object’ (first row in Figure 4.10).

By seeding half the population in this manner, these individuals will likely already have high classification fitness f_c and association fitness f_a . The randomly-generated half of the population helps to maintain diversity and explore more of the search space.

Reproduction

Genetic algorithms typically generate successive generations through a combination of genetic operators, namely crossover (also called recombination)

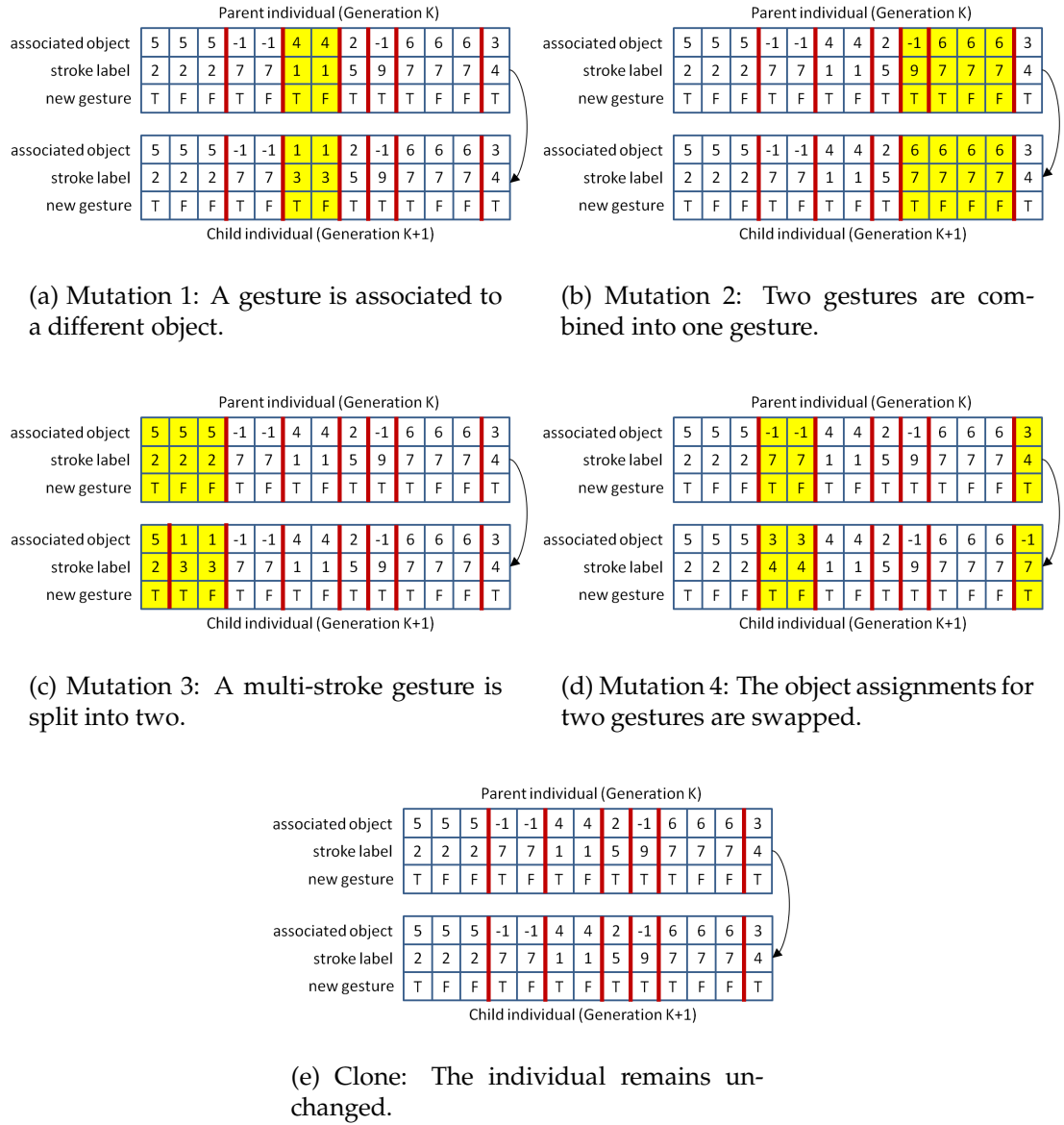


Figure 4.11: The five types of reproduction used by EAMMA to evolve the population.

and mutation. During crossover, two individuals in the current population are selected and portions of their strings are 'swapped' to produce new individuals in the next generation. However, for the genetic representation used here, crossover often produces recombinations which violate the constraint that sketches are not interspersed (i.e., users do not return to a gesture after starting

a new one). For example, if Parent A has assigned strokes 1 and 2 to a desk and Parent B has assigned stroke 7 to the same desk, any crossover operation that inherits strokes 1 and 2 from Parent A and stroke 7 from Parent B will not be a valid individual. Every candidate individual produced via crossover must be checked for validity, and in fact most will be discarded. Therefore, EAMMA does not use crossover as a mode of reproduction. Instead, cloning and four different types of mutation are used. In all cases, the parent individual from the current population is selected with probability proportional to its fitness. The five methods of reproduction are described below, and illustrated in Figure 4.11. The method of reproduction is chosen at random with probability $p_{(\cdot)}$ as indicated:

- Cloning ($p_c = 0.04$): The selected parent individual is added to the next generation unchanged.
- Mutation Type 1 ($p_{m_1} = 0.24$): A gesture (possibly consisting of more than one successive stroke) is associated to a different object, or to ‘no object’ (i.e., the object is not observed in the environment). The newly associated object must not already be associated with any other stroke(s). If the gesture is associated to ‘no object’ the gesture class is assigned randomly.
- Mutation Type 2 ($p_{m_2} = 0.24$): Two successive gestures are combined into a single gesture. The associated object for the newly combined gesture is chosen randomly between the objects of the original two gestures.
- Mutation Type 3 ($p_{m_3} = 0.24$): A multi-stroke gesture is split into two gestures. The gesture to split and the location at which it is to be split are chosen randomly. One of the two new gestures remains associated with the original object, and the other is associated to a different object (or to ‘no

object'). The newly associated object must not already be associated with any other stroke(s). If the gesture is associated to 'no object' the gesture class is assigned randomly.

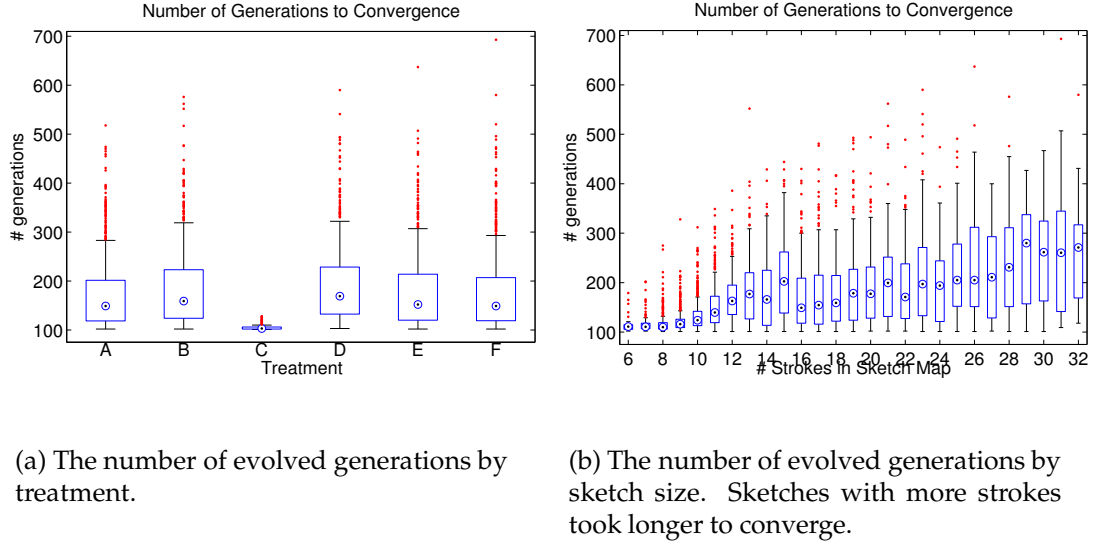
- Mutation Type 4 ($p_{m_4} = 0.24$): The object assignments for two gestures are swapped. The two gestures are selected at random.

Additionally, the 'best' individual (the individual with the highest fitness) is cloned and added to the next generation. This ensures that the best solution remains in the population and is not lost through mutation.

4.3.2 Experiments and Results

Twelve subjects provided sketch and speech navigation instructions for six maps across six phases, as described in Section 4.1.3. Sixteen trials were discarded due to insufficient data collection (the users exited the GUI before all data was properly saved), leaving 416 trials for evaluation. Since genetic algorithms are not guaranteed to converge to a global optimum, each treatment of EAMMA was performed twice for every trial. Six different variations of EAMMA were performed on the data, where each treatment can be thought of as searching for the Pareto-optimal solution in a specific direction:

- Treatment A: $F = f_p f_c f_{ma} f_a$, as described in Section 4.3.1
- Treatment B: $F = f_p f_c f_{ma}$ (no association fitness)
- Treatment C: $F = f_p f_{ma} f_a$ (no classification fitness)
- Treatment D: $F = f_c$ (only classification fitness)

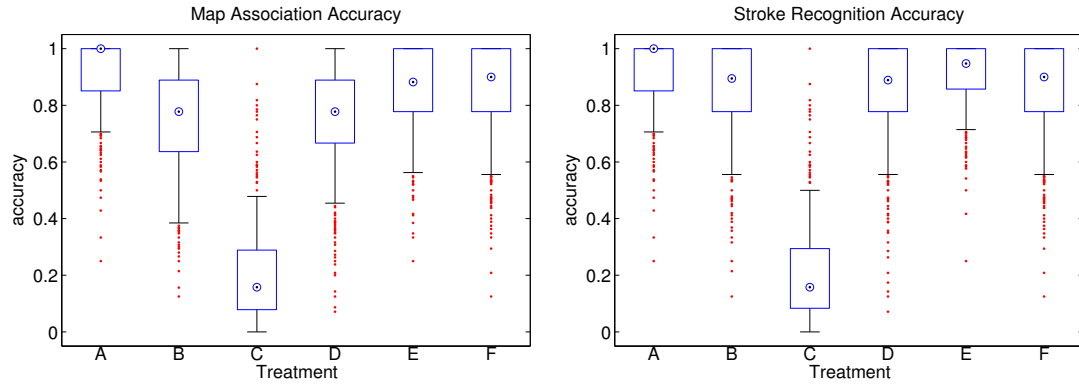


(a) The number of evolved generations by treatment. (b) The number of evolved generations by sketch size. Sketches with more strokes took longer to converge.

Figure 4.12: The number of generations evolved by EAMMA before convergence. Convergence was triggered when the best individual remained unchanged for 100 consecutive generations.

- Treatment E: $F = f_p f_c f_{ma} f_a$ where $f_a = 0$ if $N_{uo} > 0$, and 1 otherwise (i.e. assumes all observed objects in the environment are included in the sketch map, but no penalty for any ‘extra’ strokes in the sketch map not associated with an object in the environment)
- Treatment F: $F = f_p f_c f_{ma} f_a$ where $f_a = 0$ if $N_{us} > 0$, and 1 otherwise (i.e. assumes all sketched objects are observed in the environment, but no penalty for any ‘extra’ objects in the environment not associated with strokes in the sketch map)

Each trial was run with a population of size 200 for a maximum of 1000 generations. If the best solution did not improve for 100 consecutive generations, convergence was assumed and EAMMA was terminated. On average, convergence occurred after about 167 generations. However, as Figure 4.12(a) shows, the number of generations to convergence differed across treatments. As



(a) Map association accuracy by treatment. A stroke was considered correctly associated if it was assigned to the correct object (if any) in the environment.

(b) Stroke recognition accuracy by treatment. A stroke was considered correctly segmented and recognized if it was grouped into the correct gesture and assigned to the correct class.

Figure 4.13: Sketch recognition and map alignment results using EAMMA with different fitness functions.

shown in Figure 4.12(b), the number of generations increased with the number of strokes in the sketch map. This is expected, as the size of the search space is larger when there are more strokes to associate.

With respect to sketch recognition and map alignment, four performance metrics are considered. First, *stroke recognition accuracy* represents the percent of strokes in a sketch that are correctly grouped into appropriate gestures and classified as the correct class. Second, *sketch classification exactness* is the percent of complete sketch maps with all strokes correctly classified without any errors. *Map association accuracy* is the percent of strokes in the map associated to the correct object in the environment. Lastly, *sketch association exactness* is the percent of complete sketch maps with all strokes associated to the correct objects without any errors.

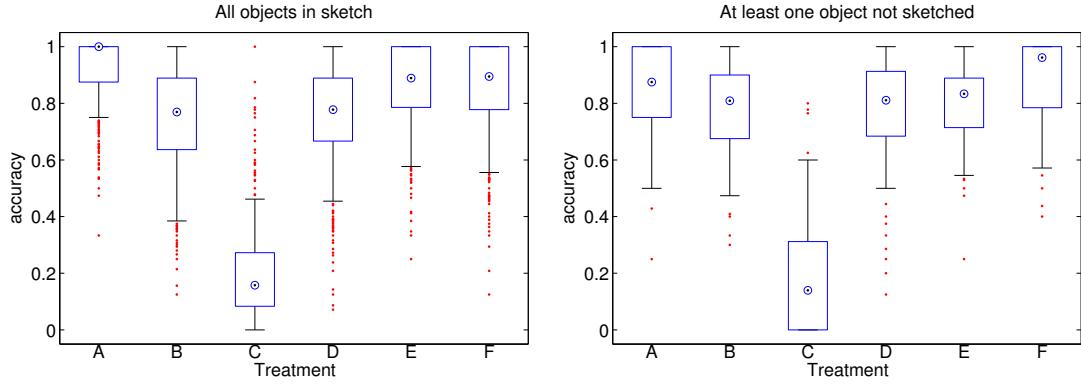
Figure 4.13(a) plots the total map association accuracy across the six treatments. For Treatment A, the average map association accuracy was 91.4%, meaning that 91% of all sketched strokes were assigned to the correct object in the environment, as intended by the user. EAMMA under Treatments E and F performed slightly worse, with an average map association accuracy around 86%. A two-sample t-test showed the difference in map association accuracy between Treatment A and Treatments E and F to be statistically significant ($t(2494) = 9.3981, p < 0.001$). Treatment E makes the assumption that all objects in the observed environment are included in the user's sketch, which was not true in 13% of cases. Treatment F assumes that every (non-path) sketched stroke corresponds to an object in the observed environment; while this was generally true, a few users sketched 'extra' gestures, such as the room boundary walls or the letter 'F' at the finish location.

Treatments B and D yielded an even lower average map association accuracy of about 75%. For both of these treatments, there was no penalty for unassociated strokes or objects. This had the effect of associating more strokes to 'no object' (even if recognized as the correct class) in order to increase the map alignment fitness f_{ma} . A two-sample t-test showed the difference in map association accuracy between Treatments E and F and Treatments B and D to be statistically significant ($t(3326) = 17.9808, p < 0.001$). Lastly, Treatment C, which assigns strokes to observed objects based only on spatial relations, performed extremely poorly with an average map association accuracy of only 20%. Not surprisingly, map-matching using point landmarks is a very challenging problem, especially when there is not a one-to-one association assumption. This confirms that the probabilistic class assignments obtained from the sketch and speech recognition phase are extremely useful for correctly aligning a sketched map with the

observed environment.

Figure 4.13(b) plots the stroke recognition accuracy across the six treatments. In this case, strokes were considered correctly segmented and recognized if its *class* was correct (e.g., `table` or `toolbox`) and it was appropriately grouped with strokes of the same gesture, regardless of the object in the environment to which it was associated (if any). For Treatments A and E, the average stroke recognition accuracy was greatest at 91%. This is higher than the 86% map association accuracy for Treatment E ($t(2494) = 11.3570, p < 0.001$), indicating that this treatment occasionally incorrectly assigned strokes to ‘no object’ while still recognizing them as the correct class. Treatments B, D and F yielded stroke recognition accuracies of about 86%, which was statistically significantly lower than that of Treatments A and E ($t(4158) = 12.5951, p < 0.001$). Treatment C again performed the poorest with an average stroke recognition accuracy of only 20%.

It should be noted that EAMMA under Treatment D solves for the highest likelihood sketch with respect to segmentation and classification, as presented in Section 4.2.2. In fact, for the same 51 maps analyzed in Section 4.2.2, the sketch recognition performance of Treatment D was nearly identical to that of the optimal solution found by the tree-search algorithm, with 88% average stroke recognition and 48% sketch classification exactness. Since the tree-search algorithm is guaranteed to find the optimal solution (in the sense that it maximizes Equation 4.2), it is promising that EAMMA is able to reproduce these results. Recall that as the number of strokes in the sketch increases, the search space for the tree-search algorithm grows exponentially. EAMMA, on the other hand, scales linearly with the number of strokes in the sketch, and is therefore a



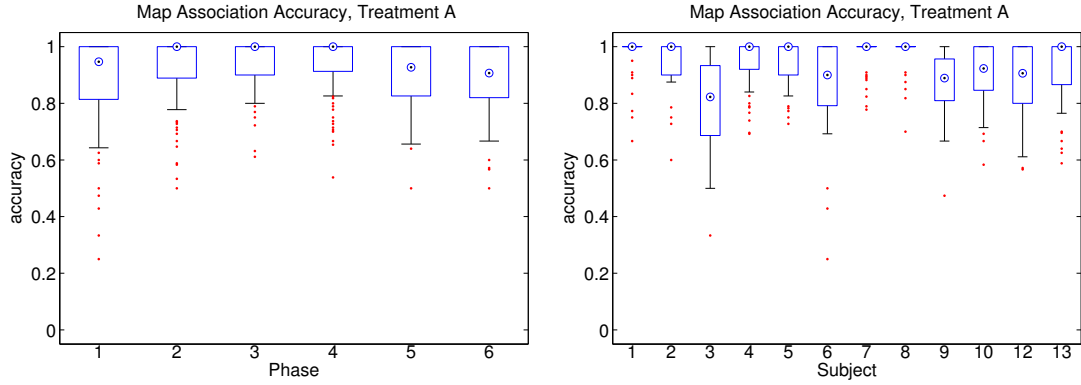
(a) Map association accuracy by treatment for cases when all objects in the environment were included in the sketch map.

(b) Map association accuracy by treatment for cases when at least one object in the environment was omitted from the sketch map.

Figure 4.14: For treatments that penalize for unassociated objects in the environment (A and E), EAMMA performs worse when the sketch map does not include all objects in the environment. This emphasizes the importance of choosing an appropriate fitness function.

practical option for sketch recognition.

Recall that one of EAMMA's assumptions is that the all (or most) of the observed objects in the environment are included in the sketch map, and is accounted for in the association fitness f_a . Figure 4.14 shows the map association accuracy for each treatment for cases in which all objects in the environment are sketched (Figure 4.14(a)) and for cases in which at least one object in the environment was not sketched (Figure 4.14(b)). In fact, 54 of the 416 sketches had at least one object omitted from the map. Treatments A and E perform slightly worse for cases in which the entire environment is not in the sketch (84% and 79% average map association accuracies, respectively), because they penalize solutions with non-associated objects via f_a . This decrease in performance is statistically significant for both Treatments A ($t(828) = 6.9826, p < 0.001$) and



(a) Map association accuracy for Treatment A across phases. EAMMA using Treatment A performs best during Phases 3–4 because stroke and gesture classification accuracy is high and sketch maps are drawn with few strokes.

(b) Map association accuracy for Treatment A across subjects.

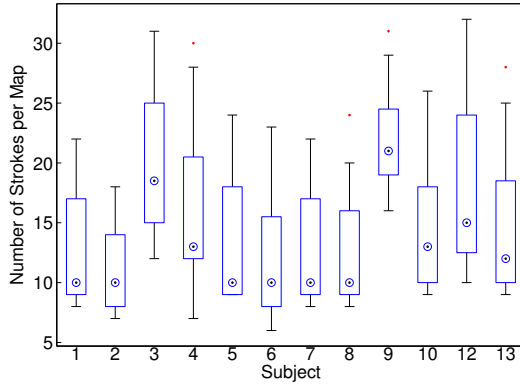
Figure 4.15: The performance of EAMMA depends on the quality and type of sketch and speech input, and is therefore not uniform across different phases and subjects.

$E(t(828) = 5.3531, p < 0.001)$. Treatments B, D, and F perform slightly better because they do not penalize non-associated objects (although these differences were not statistically significant). This emphasizes the importance of choosing an appropriate fitness function, and highlights the challenge of using a genetic algorithm for this multiobjective problem. Nevertheless, Treatment A is shown to be the most appropriate for the dataset as a whole. For the remaining analysis and discussions in this work, only Treatment A will be considered.

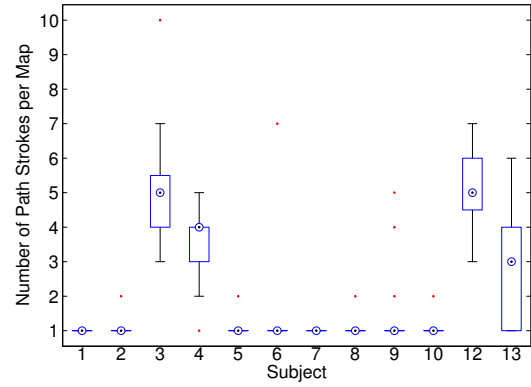
The average map association accuracy of Treatment A for each input phase is plotted in Figure 4.15(a). Here, one can see that map association is worst for Phase 1 (mean = 88%), likely because stroke and gesture classification accuracy is poorest (discussed in Section 4.2.2). Map association accuracy is highest for Phases 3 and 4 (mean = 95%), where sketched gestures are ‘simple’ and verbal

cues help further increase stroke recognition accuracy. Although gesture classification accuracy is highest for Phases 5 and 6 (see Figure 4.8(a)), the map association accuracies for these phases are lower than Phases 3 and 4 (mean = 89%, $t(560) = 6.0660, p < 0.001$). This is most likely due to the fact that sketched gestures in Phases 5 and 6 are more complicated and typically drawn with more strokes. One should expect map association to be more difficult for sketches with more strokes for two reasons. First, the size of the search space is proportional to the number of strokes in the sketch (i.e., there are more strokes to associate). Second, more strokes (especially ones located close together) provide more opportunities for strokes to be incorrectly segmented to increase the map alignment fitness term f_{ma} .

Figure 4.15(b) plots the map association accuracy across all 12 subjects for Treatment A. While most subjects' data was associated with an average accuracy $\geq 86\%$, EAMMA performed significantly worse for Subject 3 (average map association accuracy $\leq 80\%$, $t(830) = 9.0375, p < 0.001$). A likely cause for this lower map association accuracy is that Subject 3 had an accent that resulted in poor speech recognition (several grammar utterances such as "chair" and "trash can" were not reliably recognized). Another possible contribution is that the maps provided by this subject were inherently more difficult for map association. This hypothesis is partially confirmed by looking at the number of strokes used to draw each map. Figure 4.16(a) plots the number of strokes sketched per map for each subject. In general, the subjects whose maps were most poorly associated used more strokes when drawing their navigation maps. Again, since the size of the search space is proportional to the number of strokes in the sketch, these maps are inherently more difficult to perform map association using EAMMA.



(a) The number of strokes used to draw each sketch map, by subject.



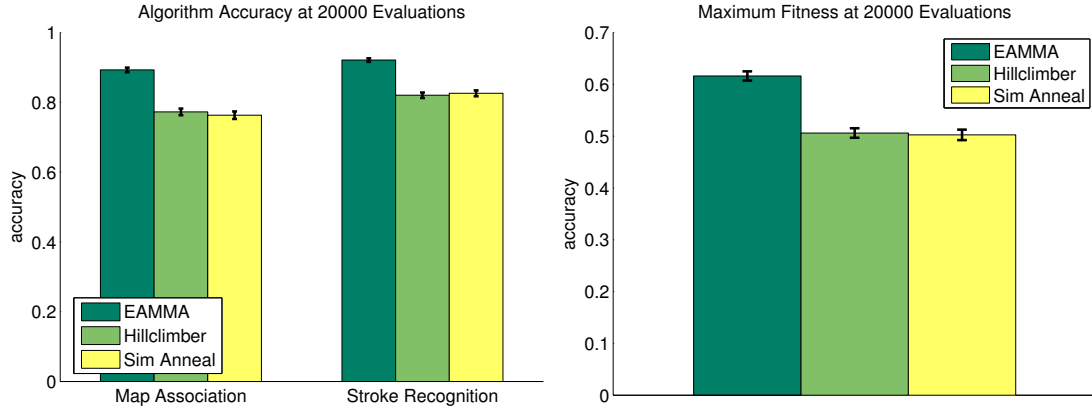
(b) The number of strokes used to draw each path, by subject. Most users sketched a single path through the map, but some drew several shorter partial-paths, resulting in a more complicated sketch to associate.

Figure 4.16: Some users (such as Subject 2) provided sketch maps with very few strokes, improving map association accuracy. Others (such as Subject 3) used many strokes, making it more difficult for EAMMA to correctly associate the sketch map.

A similar conclusion can be drawn from the results in Figure 4.16(b), which plots the number of strokes used to sketch the `path`. Most users would first sketch all the objects in the environment, and then lastly sketch a single `path` to indicate the desired trajectory. However, four subjects (Subjects 3, 4, 12 and 13) consistently sketched more than one `path` stroke. This has several implications. First, it indicates that the `path` strokes are likely shorter segments, making them more difficult to classify. Second, since `path` strokes typically weave through, between, and around objects in the map, incorrectly classifying `path` strokes as some other object class may increase the map alignment fitness term f_{ma} , especially when the sketched map is very distorted.

Figure 4.17 compares the sketch recognition results of EAMMA with those of hillclimbing and simulated annealing [100] using the fitness function defined by Treatment A. For the simulated annealing runs, the acceptance probability function $P(x_k, x_{k-1}, T_k) = \min\left(1, \exp\left\{\frac{F(x_k) - F(x_{k-1})}{T_k}\right\}\right)$, $T_0 = 1$ and $T_k = 0.999T_{k-1}$. All three algorithms were initialized using the method described in Section 4.3.1, and proposal solutions were generating using the four mutations also described in Section 4.3.1. After 20,000 fitness evaluations (200 generations with 100 individuals for EAMMA), EAMMA performed over 9% better at stroke recognition than both the hillclimber ($t(830) = 10.7930, p < 0.001$) and simulated annealing ($t(830) = 9.7441, p < 0.001$). EAMMA also performed over 12% better at map association than both the hillclimber ($t(830) = 10.4181, p < 0.001$) and simulated annealing ($t(830) = 10.3779, p < 0.001$). Figure 4.17(b) plots the average maximum fitness found by the three algorithms after 20,000 fitness evaluations. EAMMA converges to a higher fitness than both the hillclimber ($t(830) = 8.6030, p < 0.001$) and simulated annealing ($t(830) = 8.3832, p < 0.001$). These results suggest that EAMMA is better able to search the solution space than the other two algorithms, which likely get ‘stuck’ in local maxima.

In Section 4.2.2, it was hypothesized that sketch recognition could be improved by performing segmentation and classification simultaneously with map alignment. Figures 4.18(a) plots the stroke recognition accuracy for all sketches using EAMMA under Treatments A and D. Recall that Treatment D performs only segmentation and classification on the sketch, and does not consider map association at all. Figure 4.18(a) shows that sketch recognition can indeed be improved by incorporating observations of the true environment. Not only does the average stroke recognition accuracy increase by approximately 6% ($t(1662) = 8.8635, p < 0.001$), but the sketch exactness (the number of sketch



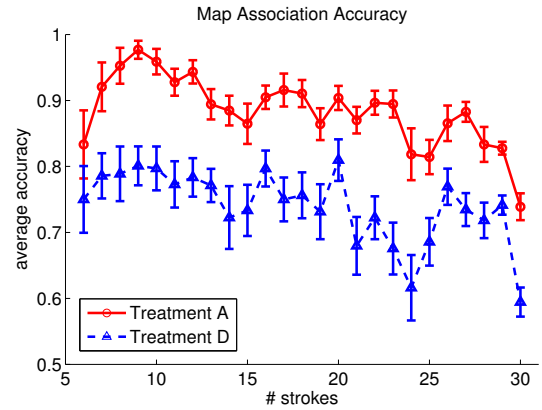
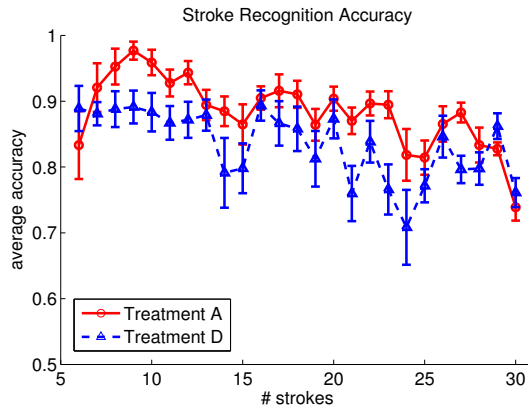
(a) Map association and stroke recognition accuracies for EAMMA, a standard hillclimber, and simulated annealing after 20000 fitness evaluations.

(b) The maximum fitness found by EAMMA, a standard hillclimber, and simulated annealing after 20000 fitness evaluations.

Figure 4.17: EAMMA converges to solutions of higher fitness values (as defined by Treatment A) than a standard hillclimber and simulated annealing, resulting in higher map association and stroke recognition accuracies.

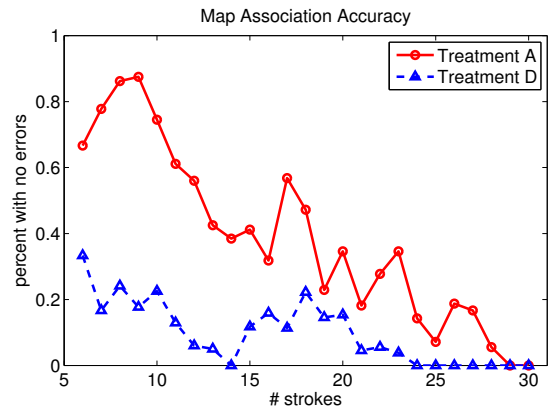
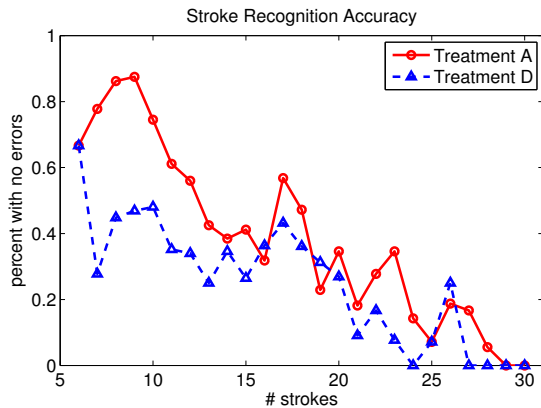
maps classified with zero errors) increases by 14%, as shown in Figure 4.18(c).

Figures 4.18(b) and (d) show an even greater difference in map association accuracy between Treatments A and D. By solving for multiple simultaneous objectives, map association accuracy increases by an average of 15% ($t(1662) = 20.5986, p < 0.001$) and sketch association exactness (the number of sketched maps correctly associated with zero errors) increases nearly 29%. These results illustrate the advantage of EAMMA over an approach such as [20] that would perform map-alignment only after objects in the sketch map have been segmentation.



(a) Average stroke recognition accuracy is improved by incorporating observations of the environment to perform simultaneous segmentation, classification and map association (Treatment A) as opposed to segmentation and classification of the sketch map alone (Treatment D).

(b) Average map association accuracy is 15% higher for Treatment A than Treatment D.



(c) The number of sketch maps segmented and classified with zero errors is about 14% higher when map association was simultaneously performed.

(d) The number of sketch maps associated to objects in the environment with zero errors is 29% higher for Treatment A than Treatment D.

Figure 4.18: EAMMA results for Treatments A and D. Plots (a) and (c) illustrate that errors in sketch recognition can be reduced by associating objects in the sketch map to those observed in the environment.

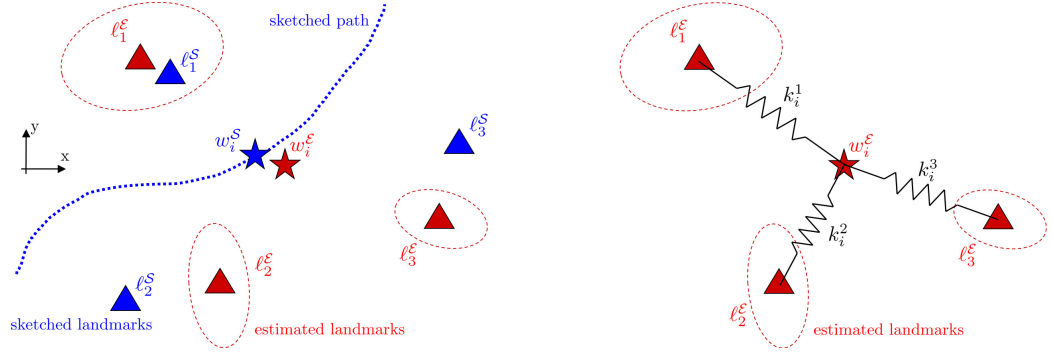
4.4 Speech Augmented Qualitative Path Planner (QPP*)

The final component to understanding and executing multimodal navigation instructions is to navigate in the true environment using the qualitative spatial relations expressed by the human. In this Section, an extension to the Qualitative Path Planner (QPP) presented in [66, 79] is proposed, which performs waypoint optimization using both spatial relations to objects as defined in the sketch map, as well as temporal relations to utterances in the user's speech. The aim is to increase navigation success while maintaining a highly flexible and natural human-robot interface. By incorporating speech as a second input mode, the user can more naturally communicate the desired path while emphasizing landmarks along the route that are most important for navigation.

4.4.1 Overview of the Qualitative Path Planner

Figure 4.19 illustrates the concepts and notation of the sketched and estimated maps, as used in the QPP. A 2D sketched map is comprised of path waypoint locations $w_i^S, i = 1 \dots n$ and landmark locations $\ell_j^S, j = 1 \dots m$. As the robot collects observations of landmarks in the true environment, an estimated map of Gaussian-distributed landmark locations $\ell_j^E = \{\bar{\ell}_j^E, \Sigma_j^E\}$ is generated. The proposed QPP calculates the location of a desired trajectory waypoint w_i^E in the estimated map/environment, corresponding to sketched waypoint w_i^S .

Since the sketched map is in error with the true representation of the environment, sketched landmark locations ℓ_j^S and estimated locations of observed landmarks in the environment $\bar{\ell}_j^E$ are not perfectly aligned with each other. In



(a) A sketch map (blue) superimposed on the estimated map (red), where the landmarks ℓ_j^S do not align perfectly with $\ell_j^\mathcal{E}$.

(b) Estimated landmarks $\ell_j^\mathcal{E}$ connected by linear springs k_i^j to the proposed waypoint location $w_i^\mathcal{E}$.

Figure 4.19: The location of waypoint $w_i^\mathcal{E}$ is influenced by surrounding landmarks $\ell_j^\mathcal{E}$ via virtual springs k_i^j . The original QPP algorithm defines k_i^j as a function of the distance between waypoint w_i^S and landmark ℓ_j^S , and the uncertainty in the estimated location of the landmark $\Sigma_j^\mathcal{E}$.

order to maintain appropriate spatial relationships within the environment, the QPP connects each waypoint $w_i^\mathcal{E}$ to each landmark $\ell_j^\mathcal{E}$ by a virtual linear spring with spring constant k_i^j , which attempts to ‘hold’ the waypoint at the same relative position as indicated on the sketched map. Each spring constant k_i^j affects the degree to which landmark j influences the location of the i^{th} estimated waypoint $w_i^\mathcal{E}$, and is calculated as a function of d_i^j , the Euclidean distance between the sketched waypoint w_i^S and sketched landmark ℓ_j^S , and the uncertainty in the location of landmark j , $\Sigma_j^\mathcal{E}$. Here, k_i^j is defined as:

$$k_i^j = \left[(d_i^j)^2 |\Sigma_j^\mathcal{E}|^{1/2} \right]^{-1} \quad (4.17)$$

In Equation 4.17, k_i^j decreases when the uncertainty in the location of landmark j is high, as specified by $\Sigma_j^\mathcal{E}$. Physically, $|\Sigma_j^\mathcal{E}(t)|^{1/2}$ is proportional to the volume of the corresponding uncertainty ellipsoid. This term discourages the

planning of trajectories relative to landmarks that haven't been localized well (or at all). Additionally, k_i^j decreases as the distance between w_i^S and ℓ_j^S increases (i.e., the location of a waypoint is more heavily influenced by closer landmarks in the sketch). The estimated location of the i^{th} waypoint $w_i^\mathcal{E}$ is found by minimizing the potential energy stored in all m springs:

$$U_i = \frac{1}{2} \sum_{j=1}^m k_i^j \|(w_i^S - \ell_j^S) - (w_i^\mathcal{E} - \bar{\ell}_j^\mathcal{E})\|_2^2 \quad (4.18)$$

The point $w_i^\mathcal{E}$ that minimizes Equation 4.18 best maintains its spatial relationships to all landmarks (proportional to k_i^j). The total potential energy in all the springs connected to the i^{th} waypoint $w_i^\mathcal{E}$ can be minimized using a quadratic program:

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} [\mathbf{w}^T H \mathbf{w} + \mathbf{c}^T \mathbf{w}] \quad (4.19)$$

where, in two dimensions:

$$\begin{aligned} \mathbf{w} = w_i^\mathcal{E} &= \begin{bmatrix} w_{i,x}^\mathcal{E} \\ w_{i,y}^\mathcal{E} \end{bmatrix} \\ H &= \begin{bmatrix} \sum_{j=1}^m k_i^j & 0 \\ 0 & \sum_{j=1}^m k_i^j \end{bmatrix} \\ \mathbf{c} &= \begin{bmatrix} 2 \sum_{j=1}^m k_i^j (\ell_{j,x}^S - w_{i,x}^S - \bar{\ell}_{j,x}^\mathcal{E}) \\ 2 \sum_{j=1}^m k_i^j (\ell_{j,y}^S - w_{i,y}^S - \bar{\ell}_{j,y}^\mathcal{E}) \end{bmatrix} \end{aligned}$$

If there are no additional constraints, a unique global solution to this optimization problem is guaranteed to exist and can be solved in closed form:

$$w_{i,x}^\mathcal{E} = \frac{\sum_{j=1}^m k_i^j (w_{i,x}^S + \bar{\ell}_{j,x}^\mathcal{E} - \ell_{j,x}^S)}{\sum_{j=1}^m k_i^j} \quad (4.20)$$

$$w_{i,y}^{\mathcal{E}} = \frac{\sum_{j=1}^m k_i^j (w_{i,y}^{\mathcal{S}} + \bar{\ell}_{j,y}^{\mathcal{E}} - \ell_{j,y}^{\mathcal{S}})}{\sum_{j=1}^m k_i^j} \quad (4.21)$$

A sensitivity study was conducted in [79], which illustrated the robustness of the QPP to inaccuracies in the sketched map and sensor range. In human trials, the QPP was shown to be an effective method for communicating navigation instructions to a mobile robot. Because the QPP mimics how humans naturally communicate route instructions, users were able to successfully interact with the system with negligible training. Due to its flexible and intuitive nature, the QPP is a promising tool for mobile robot navigation in environments for which a truth map is not available and teleoperation is not desirable. Additional details and analysis of QPP algorithm can be found in [66, 79].

4.4.2 Augmented Cost Function Using Speech Cues

The Qualitative Path Planner solves for the waypoint location $w_i^{\mathcal{E}}$ that best maintains spatial relationships to *all* landmarks in the sketched map, as specified by the relative location and uncertainty of landmarks. However, the relationship between the robot path and landmarks can be more complex, with some landmarks being more important than others. Consider the example sketch shown in Figure 4.20. Here, the blue path is drawn such that it passes exactly between two shaded landmarks. Note that there are several other objects located to the left of the path, and they will all influence the locations of the path waypoints via k_i^j , even if they are not actually important for navigation. Depending on how distorted the sketched map is with respect to the true environment, these ‘non-critical’ landmarks may actually dominate the optimization in Equation 4.19.

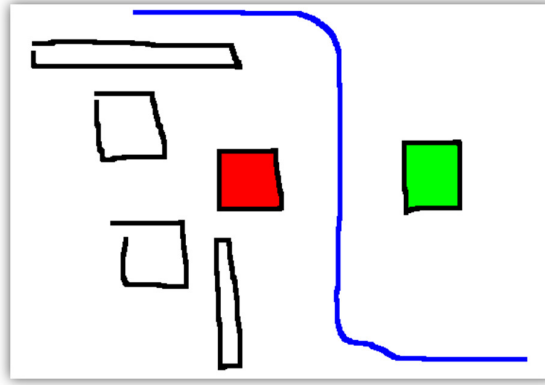


Figure 4.20: An example sketch where the path passes between two landmarks (colored for clarity). Using the original QPP algorithm, even ‘non-critical’ landmarks (unshaded objects) will influence the optimized trajectory. QPP* augments each landmarks’ influence along the path according to referenced objects in the user’s speech.

Furthermore, if one of these ‘non-critical’ landmarks is incorrectly associated, the effect on the optimized solution can be even more disastrous.

A “Speech Augmented QPP” is proposed as a richer, more robust navigation aid, where human speech can specifically be used to add relative importance information to the path planner. Specifically, if the user verbally references a landmark while sketching part of a path, it is reasonable to assume that the landmark is more ‘important’ at that point along the route. For the example shown in Figure 4.20, the user might say, “Go between the red box and the green box,” without making reference to any of the other objects in the scene. In this case, a framework is desired where the shaded landmarks should dominate the optimization problem.

The Speech Augmented QPP, referred to here as QPP*, incorporates the temporal relationship between points along the sketched path and recognized grammar utterances. For each point along the sketched path w_i^S , a new spring

constant is defined as:

$$k_i^{j(*)} = \left(\frac{T_p}{\Delta t_i^j} \right)^2 \quad (4.22)$$

where Δt_i^j is the minimum time difference between when point w_i^S was drawn and a grammar utterance corresponding to the object type of landmark j was spoken, and T_p is the total time spent drawing the full path. Here, Δt_i^j is constrained to be $\Delta t_i^j \in [2\text{sec}, T]$, which defines a two second window during which any spoken utterances are equally weighted, and prevents $k_i^{j(*)}$ growing to infinity as $\Delta t_i^j \rightarrow 0$. Only grammar utterances spoken *while sketching the path* are considered, as the user may switch between drawing the path and sketching (possibly unrelated) objects in the map.

The QPP* algorithm optimizes each waypoint location in the same fashion as QPP, with the spring constant defined as:

$$(k^{tot})_i^j = k_i^j \cdot k_i^{j(*)} \quad (4.23)$$

This formulation takes into consideration both the spatial information as indicated in the sketch, as well as temporal information from the verbal instructions. In the case where the user does not speak while drawing the path (or does not speak any grammar utterances corresponding to an object type), $(k^{tot})_i^j \rightarrow k_i^j, \forall j$ and the QPP* solution is identical to the QPP solution. This multimodal approach enables natural path planning without the need to explicitly extract *meaning* from the user's speech alone as in [80, 81, 82, 83].

4.4.3 Results and Comparison Between QPP and QPP*

Using the multimodal dataset, a few examples are presented to illustrate the difference between the original Qualitative Path Planner, QPP, and the multimodal version, QPP*. A total of 66 trials from Phase 4 were analyzed; Phase 4 was chosen due to its relatively high map association accuracy and the requirement that users provide detailed verbal instructions. For each trial, the sketch and speech inputs were first segmented, classified, and map-aligned using EAMMA, the solution of which defines which strokes in the sketch map correspond to which objects in the observed environment. The same map association solution was used by both QPP and QPP*, which were each implemented on a simulated robot in a fully-observed environment. In order to fairly compare the two algorithms, observed landmark location estimates were not updated online, such that $\Sigma_j^{\mathcal{E}}$ did not influence k_i^j in Equation 4.17. This was to ensure that the robot's trajectory did not result in different information being used by each planner.

For the 33 cases in which stroke recognition and map association were performed without any errors, there was very little difference in performance between QPP and QPP*. Some typical examples are shown in Figure 4.21. The average goal offset (distance between the desired goal point and the robot's final location) for these 33 cases was 0.72m for QPP and 0.67m for QPP*, the difference being statistically insignificant ($t(64) = 0.193, p > 0.05$). Similarly, in cases where critical errors were made during landmark association, both algorithms failed to plan sensible trajectories.

There were several instances, however, in which slight data association errors were made, yet the augmented cost function used by QPP* enabled it to compensate. Two such examples are shown in Figures 4.22 and 4.23. In both

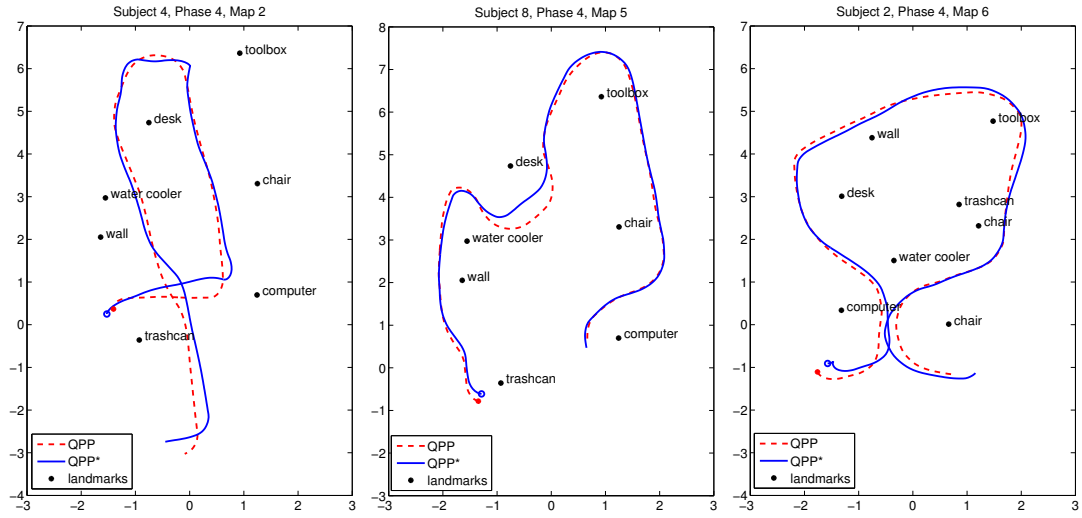
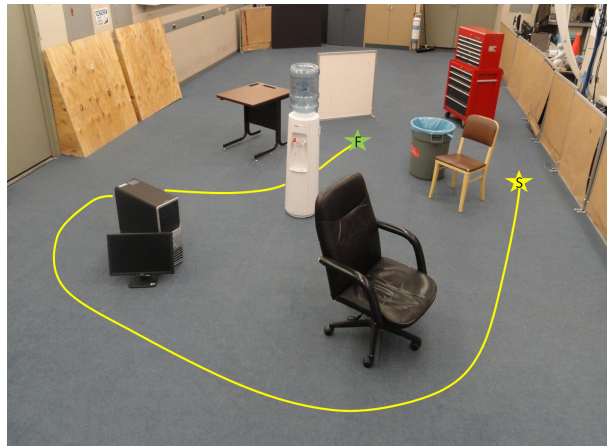


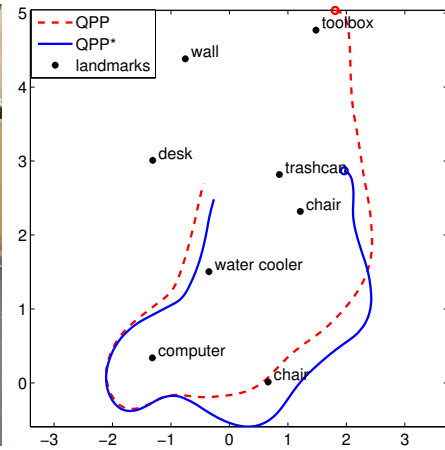
Figure 4.21: Typical examples of the trajectories planned by QPP and QPP*. In most cases where no data association errors are made between the sketch map and objects in the environment, both algorithms perform similarly.

cases, EAMMA made one or more incorrect object assignments and the original QPP algorithm planned a poor trajectory (i.e., the trajectory passes on the wrong side of a landmark). The paths planned by QPP*, however, were more successful because the users' speech allowed the planner to virtually 'ignore' these improperly associated landmarks.

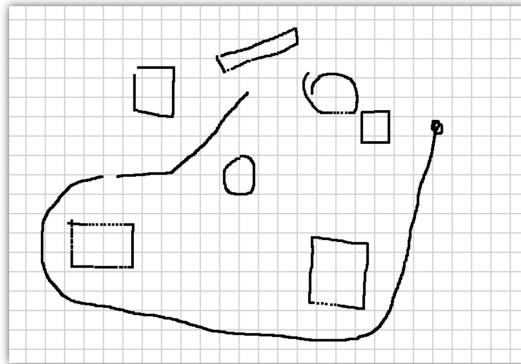
Figure 4.22(a) shows the input scene provided to the user, and 4.22(b) plots the final trajectories planned by QPP and QPP*. Figure 4.22(c) shows the user's input sketch, and 4.22(d) is the map assignment output from EAMMA. In this case, the user has not included the toolbox in the sketch map, causing the black chair in the user's sketch (lower right side) to be incorrectly assigned to the toolbox. This data association error has two significant effects on the QPP trajectory. First, it 'pushes' the start location up near where the toolbox has been observed. Second, it causes the trajectory to move on the inside of the large black chair,



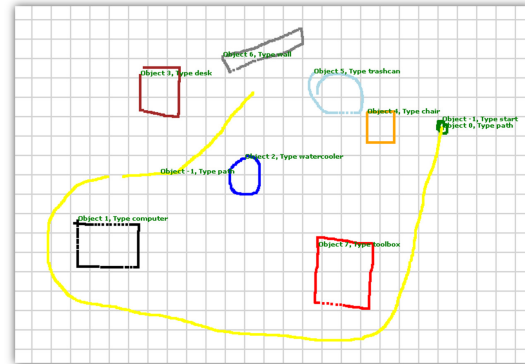
(a) Overhead image of the scene through which the robot must navigate.



(b) The corresponding trajectories planned by QPP and QPP*.



(c) The sketch map provided by the user.



(d) EAMMA assigns each stroke in the sketch map to an object in the observed environment (or to 'no object'). In this case, the black chair (lower right) has been incorrectly associated with the toolbox.

Figure 4.22: An error in data association (the sketched chair is associated with the toolbox in the environment) causes QPP to fail to plan a reasonable trajectory. QPP* is able to compensate for this error because the user makes no mention of a “toolbox” while drawing the path, so the wrongly associated landmark is virtually ‘ignored’.

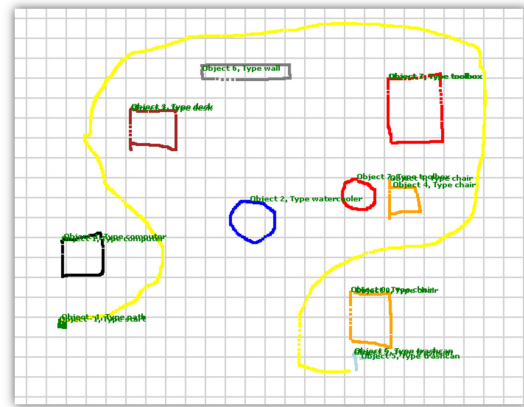
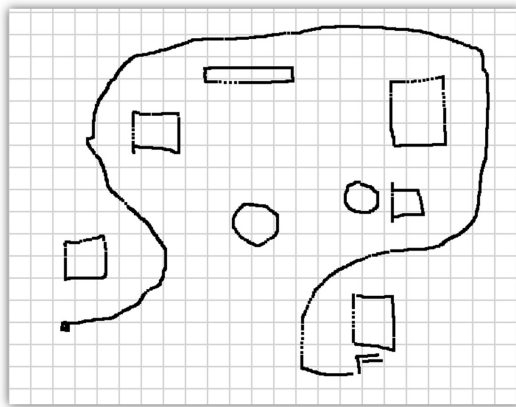
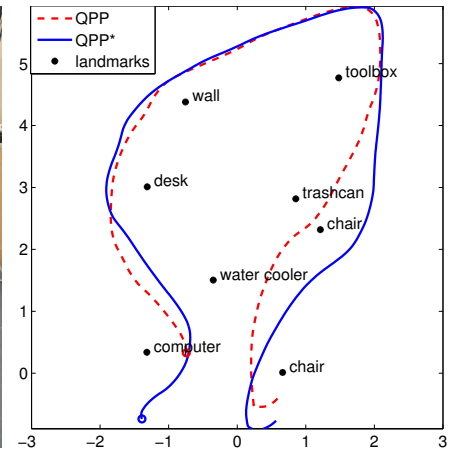
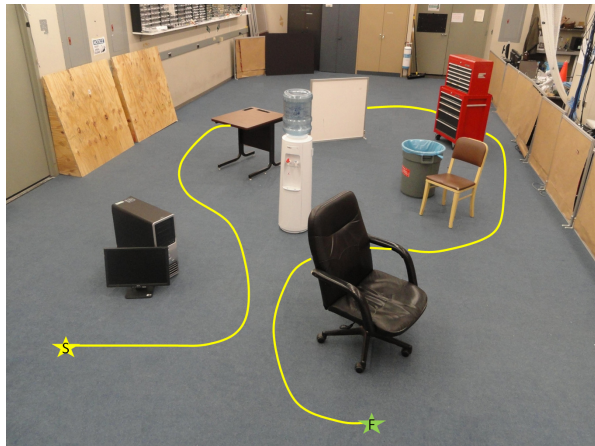
rather than around the outside.

On the contrary, the trajectory planned by QPP* starts closer to the true intended start location next to the small chair and trash can, and stays to the outside of the large black chair. The reason for this can be understood by looking at the user's speech input, below. (For clarity, speech said while the path is being sketched is shown in **bold**, and grammar utterances corresponding to landmark types are underlined.)

*"You will start here. There's a brown chair and garbage can here. A large black chair here. A black computer and monitor here. **Go down past the large black chair, around the large black chair, towards the computer and monitor, go around the black computer and monitor.** Umm, go towards a white water cooler here. A brown table is here. **Go between those, and stop in front of a white wall here.**"*

Although the user's sketched path passes nearby the black chair (which is incorrectly recognized as the toolbox), there is no mention of "toolbox" in the user's speech. However, the user twice makes reference to a "chair," a "computer" and a "monitor". This causes the spring constants associated with the computer and small brown chair in Equation 4.22 to increase, while the spring constant associated with the toolbox to be greatly decreased. In essence, the user's speech causes QPP* to 'ignore' the toolbox, even though it appears to be a spatially significant landmark.

Figure 4.23 show a similar situation. The user has indicated the finish location by sketching the letter "F," but this is an unfamiliar gesture and has not been learned by the gesture recognition algorithm. EAMMA incorrectly asso-



(c) The sketch map provided by the user.

(d) EAMMA assigns each stroke in the sketch map to an object in the observed environment (or to 'no object'). In this case, an extra sketched gesture (letter "F," lower right) has been incorrectly associated with the garbage can.

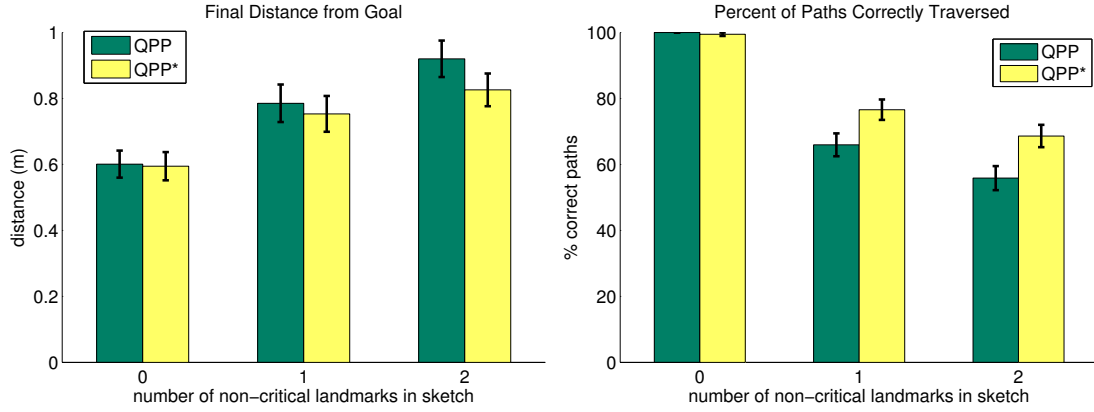
Figure 4.23: Due to an error in data association, QPP fails to plan a reasonable trajectory. QPP* is able to compensate for this error because the user makes no mention of a “trash can” while drawing the path, so the wrongly associated landmark is virtually ‘ignored’.

ciates this gesture with the trash can, and associates both the sketched garbage can and the sketched toolbox with the observed toolbox. As a result, QPP plans a path between the trash can and the small brown chair, rather than around the outside of the chair. QPP*, on the other hand, stays to the outside of the small brown chair because the user doesn't refer to the trash can in the verbal instructions:

"You'll start here near a black computer here. A black chair here. A white water cooler here. A desk here. And a white wall here. Large red toolbox, a gray garbage can and a brown chair. So start by going between the black chair and the black computer, then between the black computer and the desk. Then go around the back of the room, keeping the white wall to your right and here the red toolbox will be on your right. Then go between the brown chair and the black chair, and your destination is here."

For the last part of the path (after passing the toolbox), the user only makes reference to "chairs," and in fact the user never mentions the trash can while sketching the path. Therefore, the garbage can is virtually 'ignored' via Equation 4.22, and QPP* is able to plan a reasonable path despite this error in data association.

To illustrate how QPP and QPP* handle 'non-critical' sketched landmarks (example shown in Figure 4.20), the algorithms were run on the full dataset with the inclusion of 0–2 artificial landmarks. The artificial landmarks were assumed to be correctly associated, and their locations in the environment were randomly



(a) As the number of ‘non-critical’ landmarks increases, both QPP and QPP* perform worse, as measured by the final distance-to-goal.

(b) For sketches that include ‘non-critical’ landmarks, QPP* is better able to follow the correct path than QPP.

Figure 4.24: Performance of QPP and QPP* with the addition of simulated ‘non-critical’ landmarks in the sketch map.

generated. The artificial landmarks’ locations in the sketch maps were generated by calculating the best-fit locations, $\ell_i^S = \ell_i^E T^{E \rightarrow S}$ (where $T^{E \rightarrow S}$ is defined as in [66, 79]), and adding Gaussian noise $\mathcal{N}(0, \sigma^2)$ where $\sigma = 0.1m$. This has the effect of simulating landmarks in the user’s sketch map that aren’t referenced in the user’s speech.

A 3×2 (number of non-critical landmarks \times QPP vs. QPP*) ANOVA was used to test for differences in the final distance from goal and percent of paths correctly traversed. A path was considered to be correctly traversed if it passed all ‘critical’ landmarks (i.e., landmarks present in the original non-simulated data) on the correct side and in the appropriate direction. The number of non-critical landmarks was a significant main effect for the final goal offset ($F(2, 1122) = 5.77, p < 0.001$), and there was a significant interaction effect for the percent of paths correctly traversed ($F(2, 1122) = 3.29, p < 0.05$). Figure

4.24 plots the results of the artificial ‘non-critical’ landmark simulations. Both algorithms performed worse as the number of non-critical landmarks increased with respect to both final goal offset and path correctness. However, QPP* was better able to follow the intended paths than QPP ($t(1126) = 3.09, p < 0.01$). This is because paths planned with QPP* are more heavily influenced by landmarks that are referenced in the user’s speech, and are more likely to ‘ignore’ non-critical landmarks.

These results illustrate the capability of QPP* to improve the navigation solution by merging the effectiveness of sketch to convey qualitative spatial relations with the power of speech to disambiguate object associations. Taking advantage of the natural temporal relationships between users’ speech and sketch input, the multimodal planner is better able to recover from errors in data association and ignore non-critical landmarks than its unimodal predecessor. In combination with EAMMA, QPP* provides a robust method for humans to instruct navigation tasks to a robotic agent naturally and effectively.

4.5 Chapter Conclusions

In this chapter, a multimodal speech and sketch interface was developed for commanding navigation instructions to a mobile robot. A human user provides route directions by sketching an approximate map of the environment while giving verbal instructions, such as “go past the computer and around the chair.” Segmentation and recognition of gestures in the sketch map is framed as a probabilistic classification problem, using observation features of pixel-level data and recognized utterances from the user’s speech. By taking advantage of

learned spatial and temporal relationships between words spoken by the user and objects sketched in the map, stroke recognition accuracy is significantly improved, especially in cases where objects are sketched using ambiguous gestures. Since speech instructions do not need to be explicitly parsed for *meaning*, this multimodal approach is also more robust to ambiguities in the user's verbal instructions, despite the wide variability in users' speech and sketch styles.

In order to facilitate robot navigation, objects in the sketch map must be associated with objects in the robot's observed environment. This data association problem is complicated by the fact that 1) users do not typically draw objects to scale or in the correct relative locations, 2) users do not always draw all objects in the observed environment, 3) users occasionally draw extra gestures not corresponding to objects in the environment, and 4) a one gesture-to-one object association procedure will fail if the sketch is incorrectly segmented. To address these challenges, the Evolutionary Algorithm for Multimodal Map Association (EAMMA) is proposed for simultaneously solving the segmentation, classification, and association problems. A multiobjective fitness function is developed that aims to maximize gesture classification likelihood using observations from the sketch and speech input, maximize the similarities in spatial relations between objects in the sketch map and those observed in the environment, and minimize the number of objects in the map that are not associated to an object in the environment (and vice versa). EAMMA is shown to be more effective at interpreting the user's sketch map than performing segmentation and classification on the sketch alone.

Lastly, a method of qualitative navigation is developed, which takes advantage of the qualitative spatial relations displayed in the user's sketch map, as

well as the implied ‘importance’ of landmarks along the path based on the user’s speech. This Speech Augmented Qualitative Path Planner makes the assumption that objects are useful landmarks for navigation if they are sketched near to the path and/or referenced in the user’s speech. By incorporating multimodal input from the human, the planner is more robust to errors in map association, as ‘non-critical’ objects are not likely to be referenced by the user.

The approaches presented in this chapter create a framework for communicating navigation instructions in an efficient and natural way. Human users are able to interact with a robotic agent in a familiar way, mimicking how humans communicate with each other by using unconstrained speech and sketch. Due to the minimal amount of speech processing and the use of a probabilistic model learned from data, this interface is flexible and robust to different speech and sketching styles, and is a promising step towards truly natural human-robot interaction.

CHAPTER 5

SUMMARY

This dissertation presents several algorithms with the aim of allowing users to naturally communicate with a semi-autonomous robot using sketch and speech inputs. Drawing from prior work in handwriting recognition, Chapter 2 models sketches using a variable duration hidden Markov model with distributions learned from training data, allowing the interface to be customized to the intended users' drawing styles and shift the burden of recognition from the operator to the machine. Recognition is performed using multiclass classification techniques, and a forward search algorithm to find the combination of gestures, strokes, and interstrokes that maximizes the full sketch likelihood. Results from a set of search-and-identify missions revealed that operators were more effective at multi-tasking using the sketch interface as opposed to a more traditional button-and-menu interface, as measured by the number of satisfied mission objectives and the number of secondary tasks performed. Users reported that the biggest advantage of the sketch interface was the ability to easily perform heterogeneous tasks within a single command framework.

Chapter 3 presents a method for communicating navigation instructions to a mobile robot using an approximate sketched map, allowing the user to control the robot's path without a true map of the environment or relying on low-level teleoperation. Waypoint planning is formulated as a quadratic optimization problem which takes advantage of the probabilistic representation of the observed environment and the uncertain human input, resulting in robot trajectories in the true environment that are qualitatively similar to those provided by the human. A formal methodology is presented in which waypoints are ex-

tracted from a hand-drawn sketch, and obstacle avoidance is naturally accommodated through the addition of constraints in the optimization problem. A human trial experiment showed the planning algorithm to be robust to variations in users' sketching styles, and all ten novice users were able to successfully navigate the robot to its goal location using a freely sketched map of the environment.

Chapter 4 builds upon the concepts presented in Chapters 2 and 3 by introducing speech as a second mode of user input. Motivated by how humans communicate with one another, a multimodal interface is proposed for communicating navigation instructions using a combination of speech and sketch. The gestures drawn in the sketch map are segmented and classified using observations from pixel-level data and utterances spoken by the user. By taking advantage of learned spatial and temporal relationships between words spoken by the user and objects sketched in the map, stroke recognition accuracy is significantly improved, especially in cases where objects are sketched using ambiguous gestures. An evolutionary algorithm is developed to simultaneously perform sketch recognition and object association between the sketch map and observed environment. Once the gestures in the sketch have been recognized and associated with objects in the environment, a qualitative planner is implemented, extending the work presented in Chapter 3 to include observations extracted from the user's speech. This additional information is shown to enable some compensation for errors in data association of landmarks along the sketched route.

The methods presented in this dissertation provide a framework for future developments in natural human-robot interaction. In particular, the exhibited

success of the proposed multimodal interface motivates further examination into simpler recognition techniques by merging multiple input modes at a low level, rather than attempting to perfect the performance of any one modality. Many of the principles in this work can be applied to other input modalities and extended to various applications beyond robot navigation, opening countless new avenues in human-computer interaction to be explored. As recognition technologies, especially in the area of natural language processing, continue to advance, truly natural human-robot communication may soon become realizable.

APPENDIX A

APPENDIX

Table A.1: Features used for interstroke inference. Distances are measured in pixels.

#	Description
0	distance between stroke centroids
1	distance between last pixel of first stroke and first pixel of second stroke
2	arctangent of slope of line from last pixel of first stroke and first pixel of second stroke
3	time between finishing stroke 1 and starting stroke 2
4	distance between stroke midpoints
5	arctangent of slope of line between stroke midpoints
6	ratio of stroke lengths
7	minimum distance between strokes
8	maximum distance between strokes

Table A.2: Features used for gesture and stroke inference. (Features marked with an * are used for gestures only.) Distances are measured in pixels.

#	Name	Details/Reference
0–1	width, height	$x_{\max} - x_{\min}, y_{\max} - y_{\min}$
2	pixel count	# of pixels
3	bounding box length	[59]
4	angle of bounding box diagonal	[59]
5–6	initial, final horizontal offset	[60]
7–8	initial, final vertical offset	[60]
9	average direction	[60]
10	total time spent sketching	$t_{\max} - t_{\min}$
11	distance first-last	[60]
12*	number of strokes	
13*	total time drawing each stroke	
14*	average time drawing each stroke	
15	length	[60]
16	length down	sum of stroke lengths
17	closure	[60]
18	eccentricity	[60]
19	ratio of coordinate axes	[60]
20	mean of centroid radius	[59]
21	max angular difference	[60]
22–23	cosine, sin of initial angle	[59]
24–25	cosine, sine first and last sample	[59]
26	curvature	[60]
27	mean curvature	[60]
28	stdev of curvature	[59]
29–30	abs, squared curvature	[60]
31	perpendicularity	[60]
32	mean perpendicularity	[60]
33	stdev of perpendicularity	[59]
34	macro perpendicularity	[59]
35	mean macro perpendicularity	[59]
36	stdev of macro perpendicularity	[59]
37–38	mean, max velocity	[60]
39	stdev of velocity	[59]
40–41	mean acceleration	[60]
42	stdev of acceleration	[59]
43	max deceleration	[60]
44	HoT max bin (using 45° bins)	bin # containing most pixel tangents
45–52	global HoT shifted by max bin	bin #1 is HoT max bin
53	initial angle	atan of slope of line from first to second pixel
54–61	global HoT shifted by initial angle	bin #1 is centered at initial angle
62–69	Q1 local HoT shifted by initial angle	HoT for first 1/4 pixels
70–77	Q2 local HoT shifted by initial angle	HoT for second 1/4 pixels
78–85	Q3 local HoT shifted by initial angle	HoT for third 1/4 pixels
86–93	Q4 local HoT shifted by initial angle	HoT for fourth 1/4 pixels

Algorithm 2: Finding the most likely sketch

```

leaf nodes =  $\{\emptyset\}$ 
 $N_S$  = number of strokes in sketch
for all  $c_s \in \bar{S}$  do
    create new node  $n$ 
    set node ancestry  $A^{[n]} = \{\emptyset\}$ 
    set stroke number  $N^{[n]} = 1$ 
    set gesture number  $G^{[n]} = 1$ 
    set stroke type  $s_1^{[n]} = c_s$ 
    evaluate  $\mathcal{L}_n = p\left(s_1^{[n]} = c_s | o_{s_1}\right)$ 
    evaluate  $\mathcal{L}'_n = (\mathcal{L}_n)^{\left(\frac{N_S}{N^{[n]}}\right)^\alpha}$ 
    add  $n$  to leaf nodes
end for
find node  $m \in \text{leaf nodes}$  such that  $m = \arg \max (\mathcal{L}'_m)$ 
while  $N^{[m]} < N_S$  do
    remove  $m$  from leaf nodes
    for all  $c_i \in \bar{I}, c_s \in \bar{S}$  do
        create new node  $n$ 
        set node ancestry  $A^{[n]} = \{A^{[m]} \cup m\}$ 
        set stroke number  $N^{[n]} = N^{[m]} + 1$ 
        set stroke type  $s_{N^{[n]}}^{[n]} = c_s$ 
        set interstroke type  $i_{N^{[m]}}^{[n]} = c_i$ 
        if  $c_i$  represents a gesture self-transition then
            set gesture number  $G^{[n]} = G^{[m]}$ 
            if  $s_{N^{[n]}}^{[n]} \neq s_{N^{[n]}-1}^{[m]}$  then
                 $\mathcal{L}_n = 0$ 
            else
                evaluate  $\mathcal{L}_n = \mathcal{L}_m \cdot \mathcal{L}_{c_s}\left(s_{N^{[n]}}^{[n]}\right) \cdot \mathcal{L}_{c_i}\left(i_{N^{[m]}}^{[n]}\right)$ 
            end if
        else if  $\bar{i}_{c_i}$  represents new gesture transition then
            set gesture number  $G^{[n]} = G^{[m]} + 1$ 
            set gesture type  $g_{G^{[n]}}^{[n]} = c_g$  to match  $s_{N^{[n]}}^{[n]}$ 
            evaluate  $\mathcal{L}_n = \mathcal{L}_m \cdot \mathcal{L}_{c_s}\left(s_{N^{[n]}}^{[n]}\right) \cdot \mathcal{L}_{c_i}\left(i_{N^{[m]}}^{[n]}\right) \cdot \mathcal{L}_{c_g}\left(g_{G^{[n]}}^{[n]}\right)$ 
        end if
        evaluate  $\mathcal{L}'_n = (\mathcal{L}_n)^{\left(\frac{N_S}{N^{[n]}}\right)^\alpha}$ 
        add  $n$  to leaf nodes
    end for
    find node  $m \in \text{leaf nodes}$  such that  $m = \arg \max (\mathcal{L}'_m)$ 
end while
return  $A^{[m]}$ 

```

Table A.3: Verbal observation features o_v . Distance is measured in pixels, time measured in seconds.

Feature #	Description
0	minimum distance to Noun
1	minimum distance to Verb
2	minimum distance to Adjective
3	minimum distance to Adverb
4	minimum distance to Preposition
$5 : 4 + M$	minimum distance to each class c_g word
$5 + M$	minimum time to Noun
$6 + M$	minimum time to Verb
$7 + M$	minimum time to Adjective
$8 + M$	minimum time to Adverb
$9 + M$	minimum time to Preposition
$10 + M : 9 + 2M$	minimum time to each class c_g word
$10 + 2M$	distance overlap probability with Noun
$11 + 2M$	distance overlap probability with Verb
$12 + 2M$	distance overlap probability with Adjective
$13 + 2M$	distance overlap probability with Adverb
$14 + 2M$	distance overlap probability with Preposition
$15 + 2M : 14 + 3M$	distance overlap probability with c_g word
$15 + 3M$	time overlap probability with Noun
$16 + 3M$	time overlap probability with Verb
$17 + 3M$	time overlap probability with Adjective
$18 + 3M$	time overlap probability with Adverb
$19 + 3M$	time overlap probability with Preposition
$20 + 3M : 19 + 4M$	time overlap probability with c_g word

REFERENCES

- [1] K. Čapek. R.u.r. (rossum's universal robots), 1920.
- [2] I. Asimov. *I, Robot*. Gnome Press, 1950.
- [3] J. Badham. Short circuit, 1986.
- [4] G. Lucas. Star wars, 1977.
- [5] W. Hanna and J. Barbera. The jetsons, 1962.
- [6] D. Anderson, C. Bailey, and M. Skubic. Hidden markov model symbol recognition for sketch-based interfaces. In *AAAI Fall Symposium*, pages 15–21, Menlo Park, CA, 2004.
- [7] J. Wobbrock, A. Wilson, and Y. Li. Gestures without libraries, toolkits or training: a \$1 recognizer for user interface prototypes. In *20th Annual ACM Symposium on User Interface Software and Technology*, pages 159–168, 2007.
- [8] M. Shilman, P. Viola, and K. Chellapilla. Recognition and grouping of handwritten text in diagrams and equations. In *International Workshop on Frontiers in Handwriting Recognition*, pages 569–574, 2004.
- [9] T. Sezgin and R. Davis. Hmm-based efficient sketch recognition. In *10th International Conference on Intelligent User Interfaces*, 2005.
- [10] D. Del Vecchio, R. Murray, and P. Perona. Decomposition of human motion into dynamics based primitives with applications to drawing tasks. *Automatica*, 39:2085–2098, 2003.
- [11] C. Alvarado and R. Davis. Sketchread: A multi-domain sketch recognition engine. In *17th Annual ACM Symposium on User Interface Software and Technology*, pages 23–32, 2004.
- [12] T. Sezgin and R. Davis. Sketch recognition in interspersed drawings using time-based graphical models. *Computer Graphics*, 32:500–510, 2008.
- [13] S. Calinon and A. Billard. Stochastic gesture production and recognition model for a humanoid robot. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2004)*, volume 3, pages 2769–2774, 2004.
- [14] C. Lee and Y. Xu. Online, interactive learning of gestures for human/robot interfaces. In *IEEE International Conference on Robotics and Automation (ICRA1996)*, pages 2982–2987, 1996.
- [15] S. Waldherr, R. Romero, and S. Thrun. A gesture based interface for human-robot interaction. *Autonomous Robots*, 9:151–173, 2000.
- [16] B. Verma, M. Blumenstein, and S. Kulkarni. Recent achievements in off-line handwriting recognition systems. In *International Conference on Computational Intelligence and Multimedia Applications*, pages 27–33, 1998.
- [17] A. Corradini. Real-time gesture recognition by means of hybrid recognizers. In *Lecture Notes In Computer Science, Revised Papers from the International Gesture Workshop on Gesture and Sign Languages in Human-Computer Interaction*, volume 2298, pages 34–46, 2001.
- [18] G. Chronis and M. Skubic. Sketch-based navigation for mobile robots.

- In *12th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, volume 1, pages 284–289, 2003.
- [19] C. Bailey. *A Sketch Interface for Understanding Hand-Drawn Route Maps*. M.s. thesis, University of Missouri-Columbia, 2003.
 - [20] G. Parekh. *Scene Matching Between a Quantitative Map and a Qualitative Hand Drawn Sketch*. Ph.d. dissertation, University of Missouri-Columbia, 2007.
 - [21] V. Setalaphruk, A. Ueno, I. Kume, and Y. Kono. Robot navigation in corridor environments using a sketch floor map. In *IEEE International Symposium on Computational Intelligence in Robotics*, pages 552–557, Kobe, Japan, 2003.
 - [22] M. Skubic, C. Bailey, and G. Chronis. A sketch interface for mobile robots. In *IEEE 2003 Conference on Systems, Man and Cybernetics (SMC2003)*, pages 918–924, Washington, DC, 2003.
 - [23] M. Skubic, D. Anderson, and S. Blisard. Using a hand-drawn sketch to control a team of robots. *Autonomous Robots*, 22(4):399–410, 2007.
 - [24] G. Chronis. *Sketch-Based Navigation for Mobile Robots Using Qualitative Landmark States*. Ph.d. dissertation, University of Missouri-Columbia, 2007.
 - [25] M. Skubic, D. Anderson, M. Khalilia, and S. Kavirayani. A sketch-based interface for mulit-robot formations. In *AAAI Mobile Robot Competition 2004: Papers from the AAAI Workshops*, San Jose, CA, 2004.
 - [26] M. Maimone, A. Johnson, Y Cheng, R. Willson, and L. Matthies. Autonomous navigation results from the mars exploration rover (mer) mission. *Experimental Robotics IX*, 21, 2006.
 - [27] I. Miller, M. Campbell, D. Huttenlocher, F. Kline, A. Nathan, S. Lupashin, J. Catlin, B. Schimpf, P. Moran, N. Zych, E. Garcia, M. Kurdziel, and H. Fujishima. Team cornell’s skynet: Robust perception and planning in an urban environment. *Journal of Field Robotics – Special Issue on the 2007 DARPA Urban Challenge, Part I*, 25(8):493–527, 2008.
 - [28] P. Michon and M. Denis. When and why are visual landmarks used in giving directions? In *COSIT 2001 International Conference on Spatial Information Theory: Foundations of Geographic Information Science*, pages 292–305, 2001.
 - [29] N.M. Sgouros, G. Papakonstantinou, and P. Tsanakas. Localized qualitative navigation for indoor environments. In *IEEE 1996 International Conference on Robotics and Automation (ICRA1996)*, volume 1, pages 921–926, Minneapolis, MN, 1996.
 - [30] N. Vlassis, N. Sgouros, G. Efthivoulidis, and G. Papakonstantinou. Global path planning for autonomous qualitative navigation. In *8th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 354–359, Toulouse, France, 1996.
 - [31] R. Müller, T. Röfer, A. Lankenau, A. Musto, K. Stein, and A. Eisenkolb. Coarse qualitative descriptions in robot navigation. *Spatial Cognition II*,

- 1849:265–276, 2000.
- [32] C. Freksa, R. Moratz, and T. Barkowsky. Schematic maps for robot navigation. *Spatial Cognition II*, 1849:100–114, 2002.
 - [33] H. Shi and C. Jian and B. Krieg-Brückner. Qualitative spatial modelling of human route instructions to mobile robots. In *International Conference on Advances in Computer-Human Interactions (ACHI2010)*, 2010.
 - [34] K. Kawamura, A. Koku, D. Wikles, R. Peters II, and A. Sekmen. Toward egocentric navigation. *International Journal of Robotics and Automation*, 17(4):135–145, 2002.
 - [35] G. Chronis and M. Skubic. Robot navigation using qualitative landmark states from sketched route maps. In *2004 IEEE International Conference on Robotics and Automation (ICRA2004)*, pages 1530–1535, New Orleans, LA, 2004.
 - [36] L. Gennari, L. Kara, T. Stahovich, and K. Shimada. Combining geometry and domain knowledge to interpret hand-drawn diagrams. *Computers & Graphics*, 29:547–562, 2005.
 - [37] D. Bischel, T. Stahovich, E. Peterson, R. Davis, and A. Adler. Combining speech and sketch to interpret unconstrained descriptions of mechanical devices. In *International Joint Conference on Artificial Intelligence*, pages 1401–1406, 2009.
 - [38] O. Lemon, A. Bracy, A. Gruenstein, and S. Peters. The witas multi-modal dialogue system i. In *7th European Conference on Speech Communication and Technology*, pages 1559–1562, 2001.
 - [39] A. Correa, M. Walter, L. Fletcher, J. Glass, S. Teller, and R. Davis. Multimodal interaction with an autonomous forklift. In *5th ACM/IEEE International Conference on Human-Robot Interaction*, pages 243–250, 2010.
 - [40] M. Johnston, S. Bangalore, G. Vasireddy, A. Stent, P. Ehlen, M. Walker, S. Whittaker, and P. Maloor. Match: An architecture for multimodal dialogue systems. In *40th Annual Meeting on Association for Computational Linguistics*, pages 376–383, 2002.
 - [41] R. Bolt. “put-that-there”: Voice and gesture at the graphics interface. In *7th Annual Conference on Computer Graphics and Interactive Techniques*, volume 14, pages 262–270, 1980.
 - [42] A. Kobsa, J. Allgayer, C. Reddig, N. Reithinger, D. Schmauks, K. Karbusch, and W. Wahlster. Combining dietic gestures and natural language for referent identification. In *11th Conference on Computational Linguistics*, 1986.
 - [43] D. Perzanowski, D. Brock, M. Bugajska, S. Thomas, D. Sofge, W. Adams, S. Skubic, S. Blisard, N. Cassimatis, J. Trafton, and A. Schultz. Toward multimodal human-robot cooperation and collaboration. In *AIAA 1st Intelligent Systems Technical Conference*, Chicago, IL, 2004.
 - [44] M. Johnston, P. Cohen, D. Mcgee, S. Oviatt, J. Pittman, and I. Smith. Unification-based multimodal integration. In *35th Annual Meeting of the Association for Computational Linguistics*, pages 281–288, 1997.

- [45] P. Cohen, M. Johnston, D. McGee, S. Oviatt, J. Clow, and I. Smith. The efficiency of multimodal interaction: A case study. In *5th International Conference on Spoken Language Processing*, Sydney, Australia, 1998.
- [46] M. Johnston and S. Bangalore. Finite-state multimodal integration and understanding. *Natural Language Engineering*, 11(2):159–187, 2005.
- [47] S. Oviatt and K. Kuhn. Referential features and linguistic indirection in multimodal language. In *International Conference on Spoken Language Processing*, volume 6, pages 2339–2342, 1998.
- [48] D. Shah, J. Schneider, and M. Campbell. A robust sketch interface for natural robot control. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010)*, Taipei, Taiwan, 2010.
- [49] M. Chen, A. Kundu, and S. Srihari. Variable duration hidden markov model and morphological segmentation for handwritten word recognition. *IEEE Transactions on Image Processing*, 4:1675–1688, 1995.
- [50] W. Cho, S. Lee, and J. Kim. Modeling and recognition of cursive words with hidden markov models. *Pattern Recognition*, 28(12):1941–1953, 1995.
- [51] A. Kundu, Y. He, and M. Chen. Alternatives to variable duration hmm in handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11), 1998.
- [52] P. Burrow. *Arabic Handwriting Recognition*. M.s. thesis, University of Edinburgh, 2004.
- [53] R. Safabakhsh and P. Adibi. Nastaalign handwritten word recognition using a continuous-density variable-duration hmm. *The Arabian Journal for Science and Engineering*, 30(1B), 2005.
- [54] A. Kundu, T. Hines, B. Huyck, J. Phillips, and L. Van Guilder. Arabic handwriting recognition using variable duration hmm. In *9th International Conference on Document Analysis and Recognition*, volume 2, pages 644–648, 2007.
- [55] B. Krishnapuram, M. Figueiredo, L. Carin, and A. Hartemink. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:957–968, 2005.
- [56] D. Böhning. Multinomial logistic regression algorithm. *Annals of the Institute of Statistical Mathematics*, 44(1):197–200, 1992.
- [57] M. Tipping. The relevance vector machine. *Advances in Neural Information Processing Systems*, 12:652–658, 2000.
- [58] M. Tipping. Sparse bayesian learning and the relevance vector machine. *The Journal of Machine Learning Research*, 1:211–244, 2001.
- [59] D. Willems and R. Niels. Definitions for features used in online pen gesture recognition. Technical report, Radboud University Nijmegen, 2008.
- [60] D. Willems, R. Niels, M. Van Gerven, and L. Vuurpijl. Iconic and multi-stroke gesture recognition. *Pattern Recognition*, 42(12):3303–3312, 2009.
- [61] P. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and*

- Cybernetics*, 4(2):100–107, 1968.
- [62] R. Korf. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence*, 27:97–109, 1985.
 - [63] R. Kurnia, M. Hossain, A. Nakamura, and Y. Kuno. Generation of efficient and user-friendly queries for helper robots to detect target objects. *Advanced Robotics*, 20:499–517, 2006.
 - [64] S. Hart and L. Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. *Human Mental Workload*, 1988.
 - [65] S. Dutta. Qualitative spatial reasoning: A semi-quantitative approach using fuzzy logic. In *SSD '90 Proceedings of the First Symposium on Design and Implementation of Large Spatial Databases*, 1990.
 - [66] D. Shah and M. Campbell. A robust qualitative planner for mobile robot navigation. In *International Conference on Robotics and Automation (IRCA2011)*, Shanghai, China, 2011.
 - [67] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping (slam): Part i the essential algorithms. *Robotics and Automation Magazine*, 13:99–110, 2006.
 - [68] G. Allen. Principles and practices for communicating route knowledge. *Applied Cognitive Psychology*, 14:333–359, 2000.
 - [69] D. Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, pages 1150–1157, Corfu, Greece, 1999.
 - [70] L. Ledwich and S. Williams. Reduced sift features for image retrieval and indoor localisation. In *Australian Conference on Robotics and Automation*, 2004.
 - [71] I. Posner, P. Corke, and P. Newman. Using text-spotting to query the world. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2010)*, Taipei, Taiwan, 2010.
 - [72] C. Mandel, U. Frese, and T. Röfer. Robot navigation based on the mapping of coarse qualitative route descriptions to route graphs. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2006)*, pages 205–210, 2006.
 - [73] S. Fortune. *Voronoi diagrams and Delaunay triangulations*. Computing in Euclidean Geometry. 1992.
 - [74] D. Kalman. The most marvelous theorem in mathematics. *The Journal of Online Mathematics and Its Applications (www.JOMA.org)*, 8, 2008.
 - [75] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
 - [76] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT Press, Cambridge, 2005.
 - [77] R. Martinez-Cantin and J. A. Castellanos. Unscented slam for large-scale outdoor environments. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2005)*, Edmonton, Canada, 2005.
 - [78] D. Shah and M. Campbell. A sketch interface for robust and natural robot

- control. *Proceedings of the IEEE, Interaction Dynamics at the Interface of Humans and Smart Machines*, 2011.
- [79] D. Shah and M. Campbell. A qualitative path planner for robot navigation using human-provided maps. (*in review*), 2011.
 - [80] M. MacMahon, B. Stankiewicz, and B. Kuipers. Walk the talk: Connecting language, knowledge, and action in route instructions. In *National Conference on Artificial Intelligence*, 2006.
 - [81] T. Kollar, S. Tellex, D. Roy, and N. Roy. Toward understanding natural language directions. In *5th ACM/IEEE International Conference on Human-Robot Interaction*, pages 259–266, 2010.
 - [82] A. Huang, S. Tellex, A. Bachrach, T. Kollar, D. Roy, and N. Roy. Natural language command of an autonomous micro-air vehicle. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2663–2669, 2010.
 - [83] M. Skubic, D. Perzanowski, S. Blisard, A. Schultz, W. Adams, M. Bugajska, and D. Brock. Spatial language for human-robot dialogs. *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews*, 34(2):154–167, 2004.
 - [84] M. Salisbury, J. Hendrickson, T. Lammers, C. Fu, and S. Moody. Talk and draw: Bundling speech and graphics. *Computer*, 23(8):59–65, 1990.
 - [85] T. Nishimoto, N. Shida, T. Kobayashi, and K. Shirai. Improving human interface in drawing tool using speech, mouse and key-board. In *4th IEEE International Workshop on Robot and Human Communication, RO-MAN’95*, pages 107–112, Tokyo, Japan, 1995.
 - [86] S. Dusan, G. Gadbois, and J. Flanagan. Multimodal interaction on pda’s integrating speech and pen inputs. In *8th European Conference on Speech Communication and Technology*, pages 2225–2228, Geneva, Switzerland, 2003.
 - [87] A. Adler. *MIDOS: Multimodal Interactive Dialogue System*. Ph.d. thesis, Massachusetts Institute of Technology, 2009.
 - [88] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.
 - [89] B. Juang and L. Rabiner. Hidden markov models for speech recognition. *Technometrics*, 33(3):251–272, 1991.
 - [90] G. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 34(11):39–41, 1995.
 - [91] C. Fellbaum. Wordnet: An electronic lexical databse. Technical report, MIT Press, 1998 1998.
 - [92] L. Wang, H. Siegel, V. Roychowdhury, and A. Maciejewski. Task matching and scheduling in heterogeneous computing environments using a genetic-algorithm. *Journal of Parallel and Distributed Computing*, 47(1):8–22, 1997.
 - [93] D. Deaven and K. Ho. Molecular geometry optimization with a genetic algorithm. *Physical Review Letters*, 75(2):288–291, 1995.

- [94] J. Bongard, V. Zykov, and H. Lipson. Resilient machines through continuous self-modeling. *Science*, 314(5802):1118–1121, 2006.
- [95] D. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley Publishing Co., Reading, MA, 1989.
- [96] K. Deb. *Multiobjective Optimization Using Evolutionary Algorithms*. Wiley, Chichester, U.K., 2001.
- [97] K. Mikolajczyk, B. Leibe, and B. Schiele. Multiple object class detection with a generative model. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 26–36, 2006.
- [98] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 264–272, 2003.
- [99] G. Dorkó and C. Schmid. Object class recognition using discriminative local features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005.
- [100] S. Kirkpatrick, C. Gelatt Jr., and M. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.